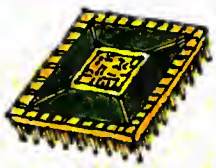
IEEE MICRO

JUNE 1993

Chips, Systems, Software, and Applications



HOT CHIPS IV

Serving up a new batch

In this issue:

- Hewlett-Packard PA7100
- DEC Alpha AXP and 21064
- Intel Pentium
- MIT Sparcle Multiprocessor

IEEE COMPUTER SOCIETY

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE MICRO

Published by the IEEE Computer Society

Volume 13 Number 3 June 1993

FEATURES

THEME ARTICLES

SPECIAL FEATURES

Cover: Jay Simpson, Design and Direction

Circulation: *IEEE Micro* (ISSN 0272-1732) is published bimonthly by the IEEE Computer Society, PO Box 3014, Los Alamitos, CA 90720-1264; IEEE Computer Society Headquarters, 1730 Massachusetts Ave., NW, Washington, DC 20036-1903; IEEE Headquarters, 345 East 47th St., New York, NY 10017. Annual subscription: $23 in addition to IEEE Computer Society or any other IEEE society member dues; $42 for members of other technical organizations. This journal is also available in microfiche form.
Postmaster: Send address changes and undelivered copies to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. Second-class postage is paid at New York, NY, and at additional mailing offices. Canadian GST#125634188.

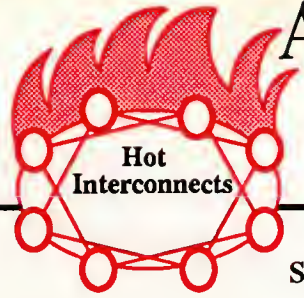
DEPARTMENTS

IEEE Computer Society
PO Box 3014, Los Alamitos, CA 90720-1264
(714) 821–8380

Editorial: Unless otherwise stated, bylined articles and descriptions of products and services reflect the author's or firm's opinion; inclusion in this publication does not necessarily constitute endorsement by the IEEE or the IEEE Computer Society. Send editorial correspondence to *IEEE Micro*, PO Box 3014, Los Alamitos, CA 90720-1264. All submissions are subject to editing for style, clarity, and space considerations.

* Submit six copies of all articles and special-issue proposals to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129 Torino, Italy; phone +39-11-564-4044; Compmail: d.delcorso; Internet: delcorso@polito.it; Bitnet: delcorso@itopoli
or
** Maurice Yunik, Dept. of Electrical Engineering, University of Manitoba, Winnipeg, Manitoba R3T 2N2, Canada; phone (204) 474-8517; yunik@eeserv.ee.umanitoba.ca

Author guidelines for article submission are available from the West Coast office (address at left).



Advance Program

Hot Interconnects'93

A Symposium on High Performance Interconnects

Sponsored by TCMM, the Technical Committee on Microprocessors and Microcomputers of the IEEE Computer Society

Stanford University, Palo Alto, California -- August 5-7, 1993

General Chair: Hasan AlKhatib, Santa Clara University

Program Co-Chairs: Paul Borrill, Sun Microsystems, and Anoop Gupta, Stanford University

Hot Interconnects is a symposium on high performance buses, multiprocessor interconnects, networks, and interfaces. Like Hot Chips, it is intended for the engineering practitioner who is interested in real solutions. Hot Interconnects is a platform for sharing knowledge and solutions among the communities of architects and designers of single systems, I/O subsystems, and networks of systems.

August 5, 1993 — Kresge Auditorium

8:45 - 9:00 **Welcome and Opening Remarks**
Hasan AlKhatib, General Chair
Paul Borrill and Anoop Gupta, Program Co-Chairs

9:00 - 10:00 **Keynote Address: Ed McCracken, President and CEO, Silicon Graphics Inc. "Interconnection Issues at Every Level in the Computer Industry: Opinions & Futures"**

10:30 - 12:00 **Advanced Bus Technology**
Session Chair: Paul Borrill, Sun Microsystems
- **"The Challenge Interconnect: Design of a 1.2 GByte per sec Coherent Multiprocessor Bus"** Mike Galles, SGI
- **"The HP suMMit Bus (PMB)"** Syrus Ziai and Cheryl Ranson, HP
- **"Electrical Design of the XDBus Using Low Voltage Swing CMOS (GTL) in the SparcCenter 2000 Server"** Christopher Cheng and Leo Yuan, Sun Microsystems
- **"LTL - A High Performance CMOS Transceiver Technology"** Bob Lipp, Consultant

12:30 - 2:00 Lunch

2:00 - 4:00 **MP Interconnects**
Session Chair: Anoop Gupta, Stanford University
- **"The Meiko CS2 Interconnect"** Mark Homewood, Meiko
- **"The SCI NodeChip, a 500 MByte per sec Virtual Backplane Chip"** Hakon Bugge and Knut Alnes, Dolphin SCI Technology
- **"The S3.mp Interconnect System and TIC Chip"** Andreas Nowatzyk and Mike Parkin, Sun Microsystems
- **"A Pipelined Network Interface for a Parallel Computer"** Gunter Watzlawik and Franz Hunter, Siemens

4:15 - 5:45 **Interconnect Design and Analysis**
Session Chair: George Cox, Intel
- **"A Cost and Speed Model for k-ary n-cube Wormhole Routers"** Andrew Chien, Univ of Illinois, Urbana-Champaign
- **"Hnet: A High-Performance Network Evaluation Testbed - Priliminary Results"** David Smitley, Frank Hady, and Dan Burns, Supercomputing Research Center
- **"Characterizing the Performance Space of Shared-Memory Machines Using Micro-Benchmarks"** Rafael Saaverda, R. Stockton Gaines, and Michael Carlton, USC

5:45 - 7:15 Reception

7:15 - 9:00 **Thursday Evening Panel: Benchmarking Interconnects**
Moderator: George Cox, Intel
- Panel Members TBD

August 6, 1993 — Kresge Auditorium

8:45 - 10:00 **Keynote Debate Moderated by Gordon Bell**
"State of the ATM Art"
Larry Roberts, NetExpress
"Is ATM the only Answer to All Networks?"
Danny Cohen, ISI

10:30 - 12:30 **High Speed Networks**
Session Chair: Van Jacobson, LBL
- **"The VIC (Virtual Interconnect with Control) Proposal for Maximizing ATM Networks Performance"** H.T. Kung, Harvard University
- **"ATM over Fibre Channel: the ANSI Solution"** Terry Anderson, Ancor
- **"Atomic: From Interconnection Network to Local Area Network"** Bob Felderman, USC/ISI
- **"Multiplexing on a High Speed Network"** Bryan Cook, IBM Poughkeepsie

12:30 - 2:00 Lunch

2:00 - 3:00 **Gigabit Links**
Session Chair: Greg Chesson, SGI
- **"The HP 2-Gbit per sec G-Link Chipset"** Chu Yen, HP
- **"Integration of Multiple Bidirectional Point to Point Serial Links in the Gbits per sec Range"** Roland Marbot, Bull S.A.

3:15 - 4:15 **New Developments**
Session Chair: Deborah Estrin, USC
- **"Hybrid Access Systems to the Internet"** Howard Strackman, Hybrid Networks
- **"New Technology in the IEEE P1394 Serial Bus - Making it Fast, Cheap and Easy to Use"** Michael Teener, Apple Computer

4:30 - 6:30 **Wireless Technologies**
Session Chair: Kathleen Nichols, Apple Computer
- **"New Markets, a New FCC, and New Spectrum"** Jim Lovette, Apple Computer
- **"Key Issues in Wireless Networking"** Chih-Lin I, AT&T Bell Labs
- **"CDMA Data Services"** Phil Karn, Qualcomm
- **"Mobile Internetworking"** Steve Deering, Xerox PARC

Organizing Committee

Publicity: Kathleen Nichols, Apple Computer, and Glen Langdon, University of California, Santa Cruz

Treasurer: Dima Khoury, TTC of Silicon Valley

Digest: Nam Ling, Santa Clara University

Registration: Qiang Li, Santa Clara University

Local Arrangements: Robert Stewart, Stewart Research

Program Committee:

Paul Borrill, Sun Microsystems
Anoop Gupta, Stanford University
Hasan AlKhatib, SCU
Dal Allen, ENDL
Gordon Bell
David Cheriton, Stanford University
Greg Chesson, Silicon Graphics
George Cox, Intel
Deborah Estrin, USC
Tom Knight, MIT
Kathleen Nichols, Apple Computer

Tutorials, Saturday, August 7, 1993.

8:30 – 12:00
Tutorial #1:
Part I: "Opto Electronics" by Schelto van Doorn (Siemens), Ron Soderstrom (IBM), and Gerry Rawls (Finisar)
Part II: "Fibre Channel" by Todd Sprenkle, Tandem

8:30 – 12:00
Tutorial #2:
"Multiprocessor Interconnects" by Andrew Chien, University of Illinois, Urbana-Champaign

12:00 – 1:30 Lunch

1:30 – 5:00
Tutorial #3:
"ATM" by Allan Fisher and Peter Steenkiste, CMU

Registration includes: attendance; two lunches; parking; one copy of notes; coffee breaks; and Thursday evening reception. On-site registration will start at 7:30 am on Thursday, August 6, 93

Registration may be faxed or e-mailed if paid by credit card.

Registration Fees for Technical Program

	Post-marked by July 12, 93	After July 12, 93
IEEE-CS or ACM Member	$200	$250
Non-Member	$250	$315
Unemployed	$180	$230
Full-Time Student	$100	$120

Registration Fees per Half-day Tutorial

	Post-marked by July 12, 93	After July 12, 93
IEEE-CS or ACM Member	$90	$105
Non-Member	$110	$135
Unemployed	$70	$85
Full-Time Student	$20	$30

Note that Hot Interconnects and Hot Chips are scheduled back-to-back.

	Thu	Fri	Sat	Sun	Mon	Tue
August	5	6	7	8	9	10
	Hot Interconnects			Hot Chips		

For more information contact Dr. Qiang Li at (408)554-2730, Fax: (408)554-5474, Internet: qli@sunrise.scu.edu

Hot Interconnects '93 Registration Form

Name _____

Affiliation _____

Mailing Address _____

_____

City State/Zip Country

Telephone No. ( ) / _____

Area Code Fax No.

Check all that apply

☐ Please, don't include my name in any mailing list.

Membership ☐ IEEE, or ☐ ACM _____ Membership No.

☐ Full-time student ☐ Unemployed

Amount paid

☐ Symposium Technical Program _____

☐ Tutorial #1 or ☐ Tutorial #2 _____

☐ Tutorial #3 _____

Payment Method: (Amount enclosed: $_____)

☐ Check drawn on a U.S. bank, ☐ VISA, ☐ MC

Make check payable to: ***Hot Interconnects Symposium***

_____

Credit Card No. Expiration Date

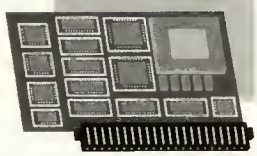
_____ / _____

Name on Credit Card **Signature**

Mail to: Dr. Qiang Li, Dept. of Computer Engr., Santa Clara University, Santa Clara, CA 95053

☐ Please, send me housing information.

On the Edge



Desktop document management

[I've asked Larry Miller, vice president of advanced products and business planning at Caere Corporation, to share with us his vision of the technologies required for desktop document management. Caere is a leading supplier of optical character recognition technology. His report follows.—S.D.]

Stephen L. Diamond

SunSoft, Inc.

Phone (415) 336-4190

Fax (415) 336-4477

steve.diamond@eng.sun.com

For quite some time, information services managers and the senior executives of businesses large and small have known about the need for document management. Existing document management solutions, however, always seem to require a total conversion of a company's established processes, a whole new set of software, and most importantly, expensive hardware upgrades.

Most MIS professionals, if just to capitalize on their current hardware and software investment, have been awaiting the day when the technology will support document management in a personal computer environment. Until quite recently, such an implementation has met significant technology barriers. Exactly what those barriers are, and how they are being solved, is the topic of this column.

Compression of images

The first barrier to document management in a PC environment has to do with the documents themselves. Most documents not under control of file handling software or add-on utilities for the operating system exist outside the computing environment. They come from outside of the company, and they start out as paper. Physically getting these paper documents into a PC environment is the first challenge.

For today's desktop scanners, the most common resolution for input is 300 dots per inch (dpi). For an 8-1/2 × 11-inch piece of paper, this calls for a full megabyte of storage per page! So the typical 80-Mbyte local hard disk of a robust system today holds just 80 pages, and more likely about 40 pages with Windows software installed. Most MIS professionals have been hoping that the technology to compress these images will soon be perfected, or that storage costs will fall significantly.

What has occurred in the storage environment is that the optical WORM (write once, read many) solutions have not expanded to the typical PC environment. While WORM devices can hold a great deal of information, their access time and cost have prevented them from becoming commonplace among PC users.

Special compression chip sets, like those created by C-Cube Microsystems, once held out some hope—their real-time compression rates were outstanding. Much of the interesting work there, however, is now devoted to video, and hence shows up in specialized multimedia systems. For all the discussion at trade shows over the past few years, we have yet to come across a mainstream PC with built-in compression technology.

Faxes are basically images, and although scanned at lower resolutions (204 × 196 or 204 × 98 dpi), they present a similar problem. When imported with a fax board, a high-resolution fax requires almost a half megabyte of memory per page. We often use the CCITT Group 3 and Group 4 compression standards in imaging applications, but the net reduction is not sufficient for real PC or local area network implementation. *[CCITT, or Comité Consultatif International de Télégraphique et Téléphonique,*

0272-1732/93/0600-004$03.00 © 1993 IEEE

is a Geneva-based standards-setting division of the International Telecommunications Union.—Ed.] These two methods operate by preserving the ratio of black and white dots. Group 3 does this on a line by line basis, and Group 4 does it bidirectionally. These produce pages of 180 to 250 Kbytes for Group 3 and about half that for Group 4. This is a major improvement, but will still not suffice.

We need more intelligent approaches to accomplish greater levels of compression in software. Caere Corporation of Los Gatos, California, has recently announced a new technology it has developed, called SuperCompression. This procedure first separates the pages into identified text and graphic areas. This small bit of information regarding the page provides huge benefits in later compression. *[For a review of compression tools, see Micro Review, p. 84.—Ed.]*

If you look closely at any photograph published in this magazine, you will see a pattern of dots; that very ratio of dark to light regions, in both directions, provides the information. However, in text areas, the ratio of black and white dots is irrelevant. Only the black dots have meaning. In fact one could throw out all the white dots! The benefit is clear. Just look at this page and imagine the percentage of the total surface area that is contained in the "white dots." By capturing the *X,Y* locations of the black dots, we have already taken a big step in compression.

Known as a leader in optical character recognition products, Caere has developed a considerable amount of technology for recognizing shapes. To further enhance the compression of text and graphics, this technology searches for redundant shapes and only saves the first example of each. This step allows compression ratios of up to 50:1 over the original image, a major improvement over existing technology. A page of text then can be saved in as little as 15 Kbytes, a level that existing magnetic storage can easily handle. Using this technology, the same 80-Mbyte drive mentioned earlier could hold over 5,000 pages.

The smaller size for images also makes transmission over a typical LAN much more practical. Pages of a megabyte each, or even 250 Kbytes, make a typical three-page document prohibitive in terms of network traffic. With only 45 Kbytes needed for the same document, LAN traffic can indeed handle document management tasks.

Optical character recognition

Absolutely key to having document management come onto desktops is the state of the OCR art. Just five years ago, the best desktop OCR technology was based on a matrix matching (matching a bitmap template with a bitmap scanned from the document). This procedure is practical only for a limited number of typewriter fonts. In the real world of laser printers, multiple fonts, and magazines, such a limitation totally inhibited the use of OCR in the office. In the last five years, OCR technology has improved greatly. Omnifont OCR, or the ability to read a wide variety of fonts and font sizes, has migrated into PCs from the world of dedicated scanning machines. Caere's OmniPage was the first software-only omnifont product. Others from Xerox, Calera, and DEST soon followed.

Omnifont OCR works much differently from matrix matching OCR. Rather than compare found shapes to a predetermined collection of shapes in memory, omnifont OCR works by analyzing the shapes for various telltale characteristics. For instance, if it determines that a character has a vertical line and a dot over the top of it, the algorithm knows it is an *i*. It does not need to know how high the line is, or how thick, or exactly the space to the dot. Only the main characteristics are needed.

Simple as this sounds, omnifont OCR is extraordinarily computation-intensive. Caere's OmniPage uses full 32-bit calculations to accomplish its work and requires over a million lines of code. Before the 386 came along, it required a coprocessor card with a CPU and memory to handle the load. Now, with 386 and even 486 systems the norm, and Windows 3.1 to provide virtual memory, the computing requirements of modern omnifont OCR are easily available.

Having the technology to input and store images, together with modern OCR that can convert the text of those images into ASCII, virtually solves input and storage problems. However, a final input-type problem remains: indexing.

Traditional imaging solutions running on minicomputers and mainframes require that the user specify keywords for each image. If you are keyboarding something and have to type the few key words, or underline them with a mouse, or add a stroke of the F1 key, it's not too much to ask. However, if the document has come in automatically, and has been supercompressed and scanned without the touch of human hands, typing in the keywords is, relatively speaking, a great deal of additional work. Needed, then, is some kind of automatic indexing to ease the input burden, which so far has merely moved downstream.

Automatic indexing

Much talk at imaging trade shows lately has focused on full text indexing, or full text retrieval. This is a handy method. However, since every word (other than the typical stop words like *or, the,* or *a*) is indexed as important, you end up finding documents that contain words, but that may not be about the words found. An article about a beauty contest and one about a *concours d'elegance* may both contain words like *sleek, beautiful,* and *stunning,* but they are about very different things.

One way to weight important words and to link relevant documents is to build a thesaurus of synonyms. This is

not as simple as including a normal thesaurus, as no normal thesaurus has a linkage for IBM, International Business Machines, and Big Blue, for example. These kinds of linkages must build upon human intelligence. Programs like Verity's Topic allow users to add linkages as they use the database; they even enable programmers or system administrators to set up such webs of relevance ahead of time. As new documents come onto the file, they pass through a filter that adds values and linkages to the special words.

Caere has taken a different approach with its new product, PageKeeper. They have come up with an algorithm that identifies the topic word of each sentence. The topic word is usually the grammatical subject of the sentence. In the earlier example of a car show and a beauty contest, the adjectives and adverbs may be similar, but the subjects to which they refer are different.

This methodology has the distinct advantage of being fully automatic and astoundingly accurate in its linkages. After all, the subjects of the sentences, when taken as a whole, are what the document is about. That's the nature of language. However, therein lies the disadvantage of this approach: it is language dependent. The rules for finding the subject of a sentence vary greatly among different languages. Each language would need a special version of the algorithm written for it. Simpler indexing schemes and lists of synonyms are not inherently variable by language.

To bring document management to the desktop, we must have some kind of automatic indexing, whether preprogrammed, added with use, or fully automatic. In combination with the new OCR and compression technologies, input of documents to a PC environment can become remarkably straightforward.

Value-added retrieval

The last of these four technical challenges is to add value to the retrieval process. If we can make the input of documents into the system as automatic as described, the barrier to entry will be low enough that a wide variety of documents will find their way into it. Imagine the variety of documents that a work group would create if all the "Please cc. (list), FYI" stickered articles and pages normally passed around by sneaker-net would just be put into the document management system. Finding such documents later and determining their relevance to the task at hand is the key.

Various approaches have been developed to accomplish this. The programmed filters will allow us to define linkages and therefore provide navigation tools through the retrieved documents. Synonym lists also provide this kind of help. But the retrieved documents come back in a list, and the document on the top of the list is not necessarily the best one. As databases grow larger, having the most relevant documents at the top becomes more important. Most readers have probably done on-line searches, but how many actually read all the retrieved material? One analyst says that in a list larger than one computer screen, most people don't read past *K*. If the best document starts with an *P*, it would never be seen.

This problem has been addressed in several ways. Some full text retrieval systems have provided a "relevance" listing that is based on the count of the found words. This helps, but long documents or bibliographies can often end up at the top without being the most relevant ones.

One company has addressed this problem by a context-based relevance weighting. Measuring the proximity of the various search words to each other in the found documents can provide a more meaningful weighting than just a rough count of found words, but it can also lead to anomalies and false positives. A car brochure mentioning an air conditioner that has passed environmental safety standards and provides quality air for the passengers might turn up as the most relevant document in a search for environmental activities last year.

Caere uses a different, more sophisticated but language-dependent methodology to rank documents in their "weighted relevance" retrieval. The algorithm weighs documents according to the degree to which search words are used as topic words, their frequency within documents, and their relative uniqueness within the database as a whole. The results are quite impressive, even "scary" according to a well-known personal computer magazine. However, since this approach is based on language, as is our understanding of the documents, such results should not be unanticipated.

With the four main technical barriers to desktop document management either down or quickly falling, we can expect to see even more solutions in this area. Presently, solutions from Lotus, with their Lotus Notes Document Imaging for enterprisewide installations, and Caere, with PageKeeper for work-group level adaptation, cover a wide spectrum of requirements. As groups and companies seek to improve their document handling and information sharing, such installations will become commonplace. It foretells a kind of *glasnost* between the worlds of paper and electronic communication, which, like the *glasnost* between countries, can only be good thing.

Send information for inclusion in Micro News one month before cover date to Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

Designers: Take note

A variety of recently announced services aim to help designers get the most out of their plans. First, consider a European microelectronics service for small- and medium-size enterprises that was announced last February.

The Chip Shop. European institutes, supported by industry, help local companies using IC technology in their products reduce costs, time to market, and risks. The Esprit III-funded shop provides regular, low-cost multiproject wafer runs for prototyping and advises on design flows and the use of ASICs. Chip Shop lets the companies capitalize on the experience of trained engineers via the JESSI and CEC Special Actions networks. Chip Shop centers can be found in Germany, France, Italy, Spain, Portugal, Greece, Denmark, Norway, and the UK.

Interested parties can contact the Chip Shop Secretariat, SCME Delft, PO Box 6067, 2600 JA Delft, The Netherlands; phone +31 15 697118 or fax +31 15 571603.

Microlithography lab. FSI International announced the opening of a microlithography applications lab in Dallas, Texas. The lab focuses on process development for 200-mm wafers, including tool characterization and enhancements, but will also support product demonstrations and address customers' process issues.

Companies can access a complete Polaris Microlithography Cluster and a Sematech GCA stepper. The Polaris alternative to conventional photolithography track systems consists of a cluster of independent modules arranged around one or two Staubli Puma 560 series clean-room robots. The modules do not require a mechanical interface. This methodology, coupled with simplicity of design, allows production systems to demonstrate high MTBFs.

For more information, contact FSI International, 322 Lake Hazeltine Drive, Chaska, MN 55318-1096; phone (612) 448-5440 or fax (612) 448-2825.

Wafer fabrication. The NCR Microelectronic Products Division will invest $81 million in an 8-inch wafer fabrication expansion in Colorado Springs, Colorado. Plans call for volume production in the third quarter of 1994. Aimed at improving process efficiencies, NCR's facility will emphasize submicron CMOS semiconductor manufacturing technologies. Call the NCR hotline at (800) 334-5454 for more information.

IC factory. Texas Instruments plans to establish a factory designed especially to provide customers with small quantities of a variety of custom circuits rather than mass-produced ICs. Consisting of integrated object-oriented software for real-time factory and process control, distributed workstations, database server, modular process systems, and single-wafer process technology, this factory promises great flexibility in wafer fabrication.

TI's Microelectronics Manufacturing Science and Technology program will produce the smaller, million-transistor circuits in near particle-free environments in fast cycle times. New program approaches include single-wafer processing, elimination of ultra clean rooms, real-time process control, easy-to-use smart equipment, and a computer-integrated manufacturing environment.

Developed by TI under contract with US government agencies, the MMST program's object-oriented microelectronics architecture also applies to other manufacturing domains.

CIM framework. Sematech has awarded a contract to Texas Instruments to codevelop and implement an industry-standard process control software framework for computer-integrated manufacturing. The standardization goal is to produce a software applications platform that will function like Microsoft Windows on a per-

0272-1732/93/0600-0007$03.00 © 1993 IEEE

sonal computer but on a much larger multicomputer, factorywide scale.

The contract also calls for enhancement of a next-generation, TI-developed application module called Specification Manager that manages manufacturing specifications and integrates into the CIM framework. Sematech calls its framework-standard application Spec_Builder.

TI and Sematech are also discussing the possibility of standardizing and enhancing six additional applications for the framework. They would be developed as part of the MMST program.

Contact John Ahearne, MS8434, Texas Instruments, 6550 Chase Oaks Blvd., Plano, TX 75023, for program information.

MCM prototyping. nChip, Inc., and the US agency, DARPA, agreed to cooperate in a development program to reduce the time and cost of developing multichip module designs.

By implementing improvements in MCM package standardization, computer-automated manufacturing, design tool interfaces, and module testing, nChip expects to reduce design turnaround times and costs by factors of four. A key goal of the 2.5-year, $5.4-million program is to reduce to less than four weeks the module development time from the hand-off of a design to nChip, to delivery of tested prototypes.

The program calls for

- a family of off-the-shelf MCM carriers to be defined and produced to eliminate tooling costs and long lead times;
- standard substrate sizes, with preprocessed power, ground, and decoupling layer, to reduce design time and allow inventorying of partially processed wafers;
- production of design kits for major EDA software tools so system designers can handle all aspects of MCM design in house; and
- a standard die library format to identify the IC information required for the design and assembly of MCMs using bare die. Logic Modeling Corporation will lead the program's die library effort and work with industry groups to help define MCM-related standards for design tool interfaces.

nChip is headquartered at 1971 North Capitol Ave., San Jose, CA 95132; phone (408) 945-9991.

PowerOpen environment. Seven manufacturers recently announced formation of an independent corporation to support their PowerOpen association founded in 1991. PowerOpen Association, Inc., will promote the PowerOpen environment and provide software developers with services that support the development of environment-based products.

Founding members Apple, Bull, Harris, IBM, Motorola, Tadpole Technology, and Thomson-CSF support the Power PC RISC architecture, Application Binary Interface specification, and a choice of user environments.

Users will be able to plan on a single version of software that runs across many systems and reduces development costs. President Dominic J. LaCava says that the association will manage a certification process that will assist end users in assessing compatibility with the PowerOpen ABI.

Further information is available from Christine Williams in Europe, phone +44 71 637-1509; and Norm Kalat in the US, phone (508) 294-4514.

Applications galore!

Today's ubiquitous micro supports a diverse list of applications; here's a sampling to keep you up to date.

SCI products. Dolphin SCI Technology AS announces its 64-bit Node Chip, which implements the IEEE Std 1596-1992 Scalable Coherent Interface in high-performance and low-cost versions. Dolphin worked closely with the SCI working group from its beginnings as the SuperBus study group in 1987, fabricated the chip at Vitesse Semiconductor, and plans to offer 200-/1,000-Mbyte/s-interconnection bandwidth versions.

Useful in shared-memory and message-passing massively parallel processor systems, workstation clustering, and high-bandwidth I/O interconnection, NodeChip implements a selected subset of SCI, incorporating transceivers, buffers, and transaction control. The single chip also supports cache-coherent systems based on the distributed shared-memory model of SCI.

The first products include a starter kit, an evaluation kit, and 500-Mbyte/s parts. Several companies are reported to have reserved a number of the chips to be fabricated in the first batch.

Intended primarily for developers, the chip's current prices and performance are not indicative of what can be expected in the longer term for high-volume applications. The DST501A Starter Kit with NodeChip, Cbus interface, key accessories, and design support lists at $19,500. An evaluation kit containing NodeChip, a VME board that behaves like a simplified SCI-VME bridge, and software costs $29,500; an additional board is $20,000. Multiple kit discounts are available.

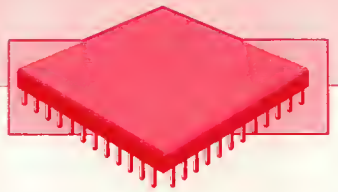
Contact Dolphin's European office via phone at +47 262 7000 or fax at +47 262 7007; scimktg@dolphin.no. The US phone and fax numbers are (603) 465-3180 and (603) 465-2680.

Wireless products. Three newly formed partnerships should ease the production of wireless products.

Communications. Sun Microsystems and two-year-old Elvis+ Ltd., a private Russian design corporation, signed a technology licensing and joint development agreement to develop wireless network communication technology. Elvis+ (short for Electronic Computer and Information Systems) will work jointly with Sun engineers on wireless communication projects and use Sun Sparcs for future workstation products.

Meter readers. Schlumberger Limited

continued on p. 102



Guest Editor's Introduction

"Hot" and "Cool" Chips

John R. Mashey

Silicon Graphics, Inc.

The annual Hot Chips Symposium is always an interesting experience. For Hot Chips IV, the program committee (chaired by Dave Patterson and myself) once again had the pleasure (and difficulty) of selecting from too many interesting papers. The conference itself was a stimulating experience, bringing together a wide variety of people from both industry and academia.

For those not familiar with Hot Chips, a brief introduction might be helpful. Hot Chips tries to offer some windows on the future, from the near future—chips readers can now buy—to futures several years away from market. To maintain this future orientation, the program committee often selects descriptions of chips that are works in progress. Some such chips are clearly research chips not intended for market. Some turn out to be research chips, although they weren't intended to be! Such uncertainty is natural when looking at the future ... so if you attend Hot Chips, please remember that not everything presented actually materializes. (Information on past symposia appears yearly in *IEEE Micro*, beginning with Hot Chips I in the April and June 1990 issues.)

Each year the symposium tries to present an interesting mixture of papers on chips that are really "hot," but in various different directions. Some hot chips really do run at high temperatures, and the program committee looks for several that stretch people's ideas of buildability. These may well be research prototypes.

Some hot chips are actually "cool"—their interest lies in their ability to pack even more performance and features into smaller and more power-efficient amounts of silicon. This category has become increasingly important.

Some hot chips are hot, not in temperature, but in terms of interest. Either they display instructive research directions, or they represent new generations of widely used chip families.

For this special issue of *IEEE Micro*, I was lucky to be able to obtain articles on the newest, high-end chips from three of the major general-purpose microprocessor families. By now, you may have seen systems based on these chips. The fourth article represents a research chip based on another major architecture. I specifically focused on high-end microprocessors to give you a good opportunity for comparison and contrast of different approaches to similar problems.

The first two articles describe aggressive implementations of existing instruction sets, one CISC, one RISC. Alpert and Avnon describe the Intel Pentium processor. This superscalar processor includes two integer pipelines and one floating-point pipeline, separate instruction and dual-ported data caches, and a branch target buffer. Many of these features are new to this architecture family, and the article analyzes the challenges of matching these features to the required architecture, especially in the area of floating point.

0272-1732/93/0600-009$03.00 © 1993 IEEE

In the second article Asprey et al. describe the Hewlett-Packard PA7100 CPU, the current fastest HP PA-RISC chip. Emphasized here are the various performance features in each of many areas of the design. You might particularly want to study the cache discussion, as HP now seems alone in choosing off-chip primary caches. It is well worth studying the reasoning.

Next McLellan describes a recent RISC architecture, Digital's Alpha AXP, and its first implementation, the 21064. This two-issue implementation emphasized high clock rates, with small on-chip caches and large off-chip secondary caches.

Finally we move forward from leading-edge, commercially available CPUs to a research project. Agarwal et al. describe Sparcle, a Sparc variation designed for research in large-scale multiprocessing. Its goal is not to build the fastest single CPU but to add mechanisms to tolerate longer memory latencies, support fine-grain parallelism, and improve interprocessor communication, all to improve large-scale multiprocessors. Thus, Sparcle illustrates solutions to the challenges of performance improvement, but in a direction orthogonal to that of the previous papers.

This business keeps moving.

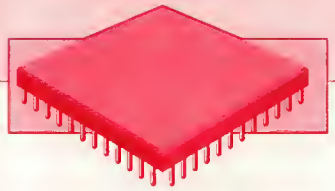
HOT CHIPS V WILL TAKE PLACE August 8-10, 1993, at Stanford University in California. The program should be available about the time of this issue's printing. If you'd like more information, contact John Hennessy at (415) 725-3712 or jlh@vsop.stanford.edu.

I thank the authors, who came through with articles on time, and the many other people who worked on this issue, especially Marie English and Dick Price of *IEEE Micro*.

John R. Mashey is director, systems technology, at Silicon Graphics, Inc. He works in a wide and rapidly changing range of technical and marketing activities. He has also worked at Bell Laboratories on various Unix-related projects and later contributed to the design of most Mips R-series RISC chips.

Mashey holds a BS degree in mathematics, and MS and PhD degrees in computer science, all from Pennsylvania State University. He served as an ACM national lecturer for four years and a Usenix program chair, and cofounded the SPEC benchmarking group. He has given more than 400 public talks on Unix, the Programmer's Workbench, software engineering, benchmarking, and the RISC architecture. He is a member of the IEEE Computer Society and the Association of Computing Machinery.

Address questions concerning this special issue to John R. Mashey, Silicon Graphics 7U-005, 2011 North Shoreline Blvd., Mountain View, CA 94039; mash@sgi.com.



Architecture of the Pentium Microprocessor

The Pentium CPU is the latest in Intel's family of compatible microprocessors. It integrates 3.1 million transistors in 0.8-μm BiCMOS technology. We describe the techniques of pipelining, superscalar execution, and branch prediction used in the microprocessor's design.

Donald Alpert

Dror Avnon

Intel Corporation

The Pentium processor is Intel's next generation of compatible microprocessors following the popular i486 CPU family. The design started in early 1989 with the primary goal of maximizing performance while preserving software compatibility within the practical constraints of available technology. The Pentium processor integrates 3.1 million transistors in 0.8-µm BiCMOS technology and carries the Intel trademark. We describe the architecture and development process employed to achieve this goal.

Technology

The continual advancement of semiconductor technology promotes innovation in microprocessor design. Higher levels of integration, made possible by reduced feature sizes and increased interconnection layers, enable designers to deploy additional hardware resources for more parallel computation and deeper pipelining. Faster device speeds lead to higher clock rates and consequently to requirements for larger and more specialized on-chip memory buffers.

Table 1 (next page) summarizes the technology improvements associated with our three most recent microprocessor generations. The 0.8-µm BiCMOS technology of the Pentium microprocessor enables 2.5 times the number of transistors and twice the clock frequency of the original i486 CPU, which was implemented in 1.0-µm CMOS.

Compatibility

Since introduction of the 8086 microprocessor in 1978, the X86 architecture has evolved through several generations of substantial functional enhancements and technology improvements, including the 80286 and i386 CPUs. Each of these CPUs was supported by a corresponding floating-point unit. The i486 CPU,[1] introduced in 1989, integrates the complete functionality of an integer processor, floating-point unit, and cache memory into a single circuit.

The X86 architecture greatly appealed to software developers because of its widespread application as the central processor of IBM-compatible personal computers. The success of the architecture in PCs has in turn made the X86 popular for commercial server applications as well. Figure 1 shows some of the well-known software environments that are hosted on the architecture.

The common software environments allow the X86 architecture to exercise several operating modes. Applications developed for DOS use 16-bit real mode (or virtual 8086 mode) and MS Windows. Early versions of OS/2 use 16-bit protected mode, and applications for other popular environments use 32-bit flat (unsegmented) mode. The Pentium microprocessor employs general techniques for improving performance in all operating modes, as well as certain techniques for improving performance in specific operating

0272-1732/93/0600-0011$03.00 © 1993 IEEE

Table 1. Technology for microprocessor development.

Microprocessor	Year	Technology	No. of transistors	Frequency (MHz)
i386 CPU	1986	1.5-µm CMOS, two-layer metal	275K	16
i486 CPU	1989	1.0-µm CMOS, two-layer metal	1.2M	33
Pentium CPU	1993	0.8-µm BiCMOS, three-layer metal	3.1M	66

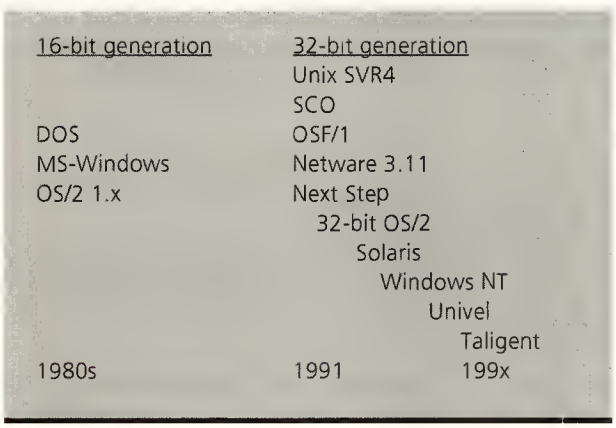


Figure 1. Software environments. *(All figures, tables, and photographs published in this article are the property of Intel Corporation.)*

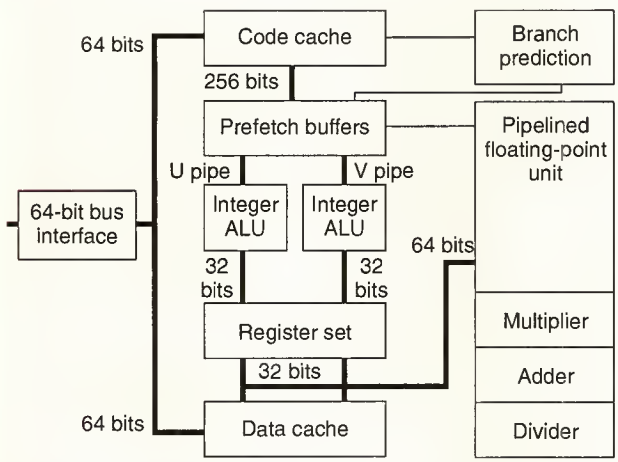


Figure 2. Pentium processor block diagram.

modes. We focus on the 32-bit flat mode here, since this is the most appropriate mode for comparison with the other high-performance microprocessors described at the Hot Chips IV Conference.

The X86 architecture supports the IEEE-754 standard for floating-point arithmetic.[2] In addition to required operations on single-precision and double-precision formats, the X86 floating-point architecture includes operations on 80-bit, extended-precision format and a set of basic transcendental functions.

Pentium CPU designers found numerous exciting technical challenges in developing a microarchitecture that maintained compatibility with such a diverse software base. Later in this article we present examples of techniques for supporting self-modifying code and the stack-oriented, floating-point register file.

Performance

A microprocessor's performance is a complex function of many parameters that vary between applications, compilers, and hardware systems. In developing the Pentium microprocessor, the design team addressed these aspects for each of the popular software environments. As a result, Pentium CPU features tuned compilers and cache memory.

We focus on the performance of SPEC benchmarks for both the Pentium microprocessor and i486 CPU in systems with well-tuned compilers and cache memory. More specifically, the Pentium CPU achieves roughly two times the speedup on integer code and up to five times the speedup on floating-point vector code when compared with an i486 CPU of identical clock frequency.

Organization

Figure 2 shows the overall organization of the Pentium microprocessor. The core execution units are two integer pipelines and a floating-point pipeline with dedicated adder, multiplier, and divider. Separate on-chip instruction code and data caches supply the memory demands of the execution units, with a branch target buffer augmenting the instruction cache for dynamic branch prediction. The external interface includes separate address and 64-bit data buses.

Integer pipeline

The Pentium processor's integer pipeline is similar to that of the i486 CPU.[3] The pipeline has five stages (see Figure 3) with the following functions:

- *Prefetch.* During the PF stage the CPU prefetches code from the instruction cache and aligns the code to the

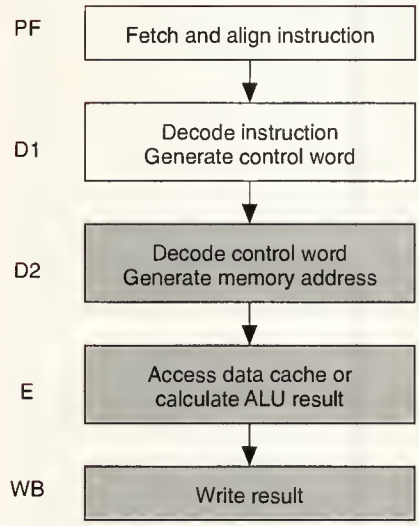


Figure 3. Integer pipeline.

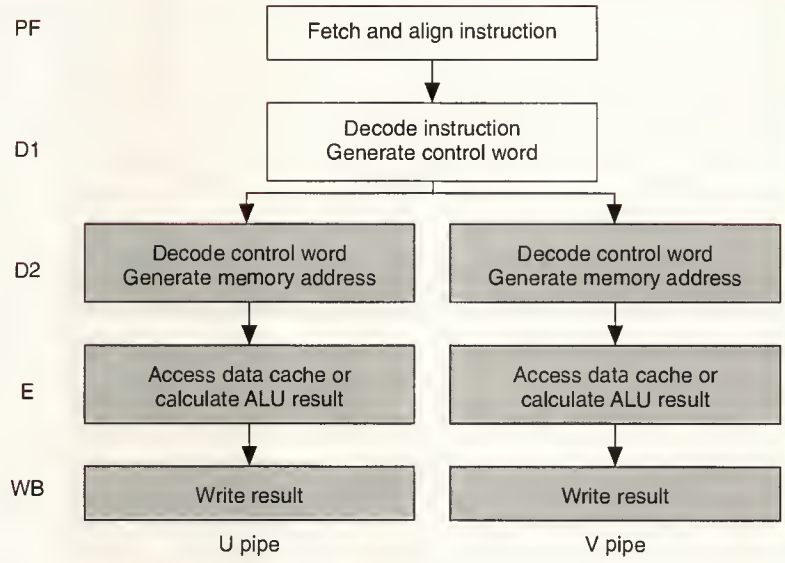


Figure 4. Superscalar execution.

initial byte of the next instruction to be decoded. Because instructions are of variable length, this stage includes buffers to hold both the line containing the instruction being decoded and the next consecutive line.

- *First decode.* In the D1 stage the CPU decodes the instruction to generate a control word. A single control word executes instructions directly; more complex instructions require microcoded control sequencing in D1.
- *Second decode.* In the D2 stage the CPU decodes the control word from D1 for use in the E stage. In addition, the CPU generates addresses for data memory references.
- *Execute.* In the E stage the CPU either accesses the data cache or calculates results in the ALU (arithmetic logic unit), barrel shifter, or other functional units in the data path.
- *Write back.* In the WB stage the CPU updates the registers and flags with the instruction's results. All exceptional conditions must be resolved before an instruction can advance to WB.

Compared to the integer pipeline of the i486 CPU, the Pentium microprocessor integrates additional hardware in several stages to speed instruction execution. For example, the i486 CPU requires two clocks to decode several instruction formats, but the Pentium CPU takes one clock and executes shift and multiply instructions faster. More significantly, the Pentium processor substantially enhances superscalar execution, branch prediction, and cache organization.

Superscalar execution. The Pentium CPU has a superscalar organization that enables two instructions to execute in parallel. Figure 4 shows that the resources for address generation and ALU functions have been replicated in independent integer pipelines, called U and V. (The pipeline names were selected because U and V were the first two consecutive letters of the alphabet neither of which was the initial of a functional unit in the design partitioning.) In the PF and D1 stages the CPU can fetch and decode two simple instructions in parallel and issue them to the U and V pipelines. Additionally, for complex instructions the CPU in D1 can generate microcode sequences that control both U and V pipelines.

Several techniques are used to resolve dependencies between instructions that might be executed in parallel. Most of the logic is contained in the instruction issue algorithm (see Figure 5) of D1.

```
Decode two consecutive instructions: I1 and I2
If the following are all true
        I1 is a "simple" instruction
        I2 is a "simple" instruction
        I1 is not a jump instruction
        Destination of I1 ≠ source of I2
        Destination of I1 ≠ destination of I2
Then issue I1 to U pipe and I2 to V pipe
Else issue I1 to U pipe
```

Figure 5. Instruction issue algorithm.

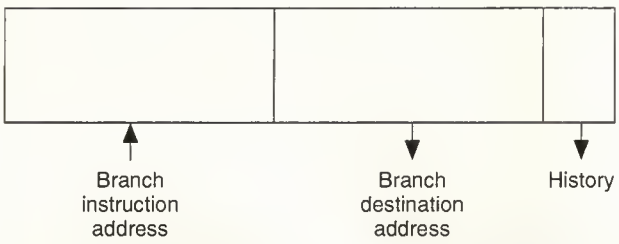


Figure 6. Branch target buffer.

Resource dependencies. A resource dependency occurs when two instructions require a single functional unit or data path. During the D1 stage, the CPU only issues two instructions for parallel execution if both are from a class of "simple" instructions, thereby eliminating most resource dependencies. The instructions must be directly executed, that is, not require microcode sequencing. The instruction being issued to the V pipe can be an ALU operation, memory reference, or jump. The instruction being issued to the U pipe can be from the same categories or from an additional set that uses a functional unit available only in the U pipe, such as the barrel shifter. Although the set of instructions identified as "simple" might seem restrictive, more than 90 percent of instructions executed in the Integer SPEC benchmark suite are simple.

Data dependencies. A data dependency occurs when one instruction writes a result that is read or written by another instruction. Logic in D1 ensures that the source and destination registers of the instruction issued to the V pipe differ from the destination register of the instruction issued to the U pipe. This arrangement eliminates read-after-write (RAW) and write-after-write (WAW) dependencies. Write-after-read (WAR) dependencies need not be checked because reads occur in an earlier stage of the pipelines than writes.

The design includes logic that enables instructions with certain special types of data dependency to be executed in parallel. For example, a conditional branch instruction that tests the flag results can be executed in parallel with a compare instruction that sets the flags.

Control dependencies. A control dependency occurs when the result of one instruction determines whether another instruction will be executed. When a jump instruction is issued to the U pipe, the CPU in D1 never issues an instruction to the V pipe, thereby eliminating control dependencies.

Note that resource dependencies and data dependencies between memory references are not resolved in D1. Dependent memory references can be issued to the two pipelines; we explain their resolution in the description of the data cache.

Branch prediction. The i486 CPU has a simple technique for handling branches. When a branch instruction is executed, the pipeline continues to fetch and decode instructions along the sequential path until the branch reaches the E stage. In E, the CPU fetches the branch destination, and the pipeline resolves whether or not a conditional branch is taken. If the branch is not taken, the CPU discards the fetched destination, and execution proceeds along the sequential path with no delay. If the branch is taken, the fetched destination is used to begin decoding along the target path with two clocks of delay. Taken branches are found to be 15 percent to 20 percent of instructions executed, representing an obvious area for improvement by the Pentium processor.

The Pentium CPU employs a branch target buffer (BTB), which is an associative memory used to improve performance of taken branch instructions (see Figure 6). When a branch instruction is first taken, the CPU allocates an entry in the branch target buffer to associate the branch instruction's address with its destination address and to initialize the history used in the prediction algorithm. As instructions are decoded, the CPU searches the branch target buffer to determine whether it holds an entry for a corresponding branch instruction. When there is a hit, the CPU uses the history to determine whether the branch should be taken. If it should, the microprocessor uses the target address to begin fetching and decoding instructions from the target path. The branch is resolved early in the WB stage, and if the prediction was incorrect, the CPU flushes the pipeline and resumes fetching along the correct path. The CPU updates the dual-ported history in the WB stage. The branch target buffer holds entries for predicting 256 branches in a four-way associative organization.

Using these techniques, the Pentium CPU executes correctly predicted branches with no delay. In addition, conditional branches can be executed in the V pipe paired with a compare or other instruction that sets the flags in the U pipe. Branching executes with full compatibility and no modification to existing software. (We explain aspects of interactions between branch prediction and self-modifying code later.)

Cache organization. The i486 CPU employs a single on-chip cache that is unified for code and data. The single-ported cache is multiplexed on a demand basis between sequential code prefetches of complete lines and data references to individual locations. As just explained, branch targets are prefetched in the E stage, effectively using the same hardware as data memory references. There are potential advantages for such an organization over one that separates code and data.

1) For a given size of cache memory, a unified cache has a higher hit rate than separate caches because it balances the total allocation of code and data lines automatically.
2) Only one cache needs to be designed.
3) Handling self-modifying code can be simpler.

Despite these potential advantages of a unified cache, all of which apply to the i486 CPU, the Pentium microprocessor uses separate code and data caches. The reason is that the superscalar design and branch prediction demand more bandwidth than a unified cache similar to that of the i486 CPU can provide. First, efficient branch prediction requires that the destination of a branch be accessed simultaneously with data references of previous instructions executing in the pipeline. Second, the parallel execution of data memory references requires simultaneous accesses for loads and stores. Third, in the context of the overall Pentium microprocessor design, handling self-modifying code for separate code and data caches is only marginally more complex than for a unified cache.

The instruction cache and data cache are each 8-Kbyte, two-way associative designs with 32-byte lines.

Programs executing on the i486 CPU typically generate more data memory references than when executing on RISC microprocessors. Measurements on Integer SPEC benchmarks show 0.5 to 0.6 data references per instruction for the i486 CPU[4] and only 0.17 to 0.33 for the Mips processor.[5] This difference results directly from the limited number (eight) of registers for the X86 architecture, as well as procedure-calling conventions that require passing all parameters in memory. A small data cache is adequate to capture the locality of the additional references. (After all, the additional references have sufficient locality to fit in the register file of the RISC microprocessors.) The Pentium microprocessor implements a data cache that supports dual accesses by the U pipe and V pipe to provide additional bandwidth and simplify compiler instruction scheduling algorithms.

Figure 7 shows that the address path to the translation look-aside buffer and data cache tags is a fully dual-ported structure. The data path, however, is single ported with eight-way interleaving of 32-bit-wide banks. When a bank conflict occurs, the U pipe assumes priority, and the V pipe stalls for a clock cycle. The bank conflict logic also serves to eliminate data dependencies between parallel memory references to a single location. For memory references to double-precision floating-point data, the CPU accesses consecutive banks in parallel, forming a single 64-bit path.

The design team considered a fully dual-ported structure for the data cache, but feasibility studies and performance simulations showed the interleaved structure to be more effective. The dual-ported structure eliminated bank conflicts, but the SRAM cell would have been larger than the cell used in the interleaved scheme, resulting in a smaller cache and lower hit ratio for the allocated area. Additionally, the handling of data dependencies would have been more complex.

With a write-through cache-consistency protocol and 32-bit data bus, the i486DX2 CPU uses buses 80 percent of the time; 85 percent of all bus cycles are writes. (The i486DX2 CPU has a core pipeline that operates at twice the bus clock's frequency.) For the Pentium microprocessor, with its higher performance core pipelines and 64-bit data bus, using a write-back protocol for cache consistency was an obvious enhancement. The write-back protocol uses four states: modified, exclusive, shared, and invalid (MESI).

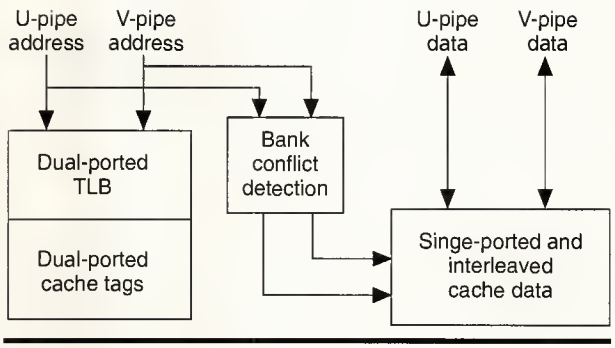


Figure 7. Dual-access data cache.

Self-modifying code. One challenging aspect of the Pentium microprocessor's design was supporting self-modifying code compatibly. Compatibility requires that when an instruction is modified followed by execution of a taken branch instruction, subsequent executions of the modified instruction must use the updated value. This is a special form of dependency between data stores and instruction fetches.

The interaction between branch predictions and self-modifying code requires the most attention. The Pentium CPU fetches the target of a taken branch before previous instructions have completed stores, so dedicated logic checks for such conditions in the pipeline and flushes incorrectly fetched instructions when necessary. The CPU thoroughly verifies predicted branches to handle cases in which an instruction entered in the branch target buffer might be modified. The same mechanisms used for consistency with external memory maintain consistency between the code cache and data cache.

Floating-point pipeline

The i486 CPU integrated the floating-point unit (FPU) on chip, thus eliminating overhead of the communication protocol that resulted from using a coprocessor. Bringing the FPU on chip substantially boosted performance in the i486 CPU. Nevertheless, due to limited devices available for the FPU, its microarchitecture was based on a partial multiplier array and a shift-and-add data path controlled by microcode. Floating-point operations could not be pipelined with any other floating-point operations; that is, once a floating-point instruction is invoked, all other floating-point instructions stall until its completion.

The larger transistor budget available for the Pentium microprocessor permits a completely new approach in the design of the floating-point microarchitecture. The aggressive

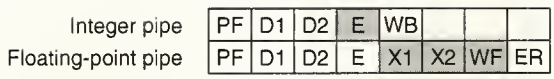


Figure 8. Floating-point pipeline.

performance goals for the FPU presented an exciting challenge for the designers, even with more silicon resources available. Furthermore, maintaining full compatibility with previous products and with the IEEE standard for floating-point arithmetic was an uncompromising requirement.

Floating-point pipeline stages. Pentium's floating-point pipeline consists of eight stages. The first two stages are processed by the common (integer pipeline) resources for prefetch and decode. In the third stage the floating-point hardware begins activating logic for instruction execution. All of the first five stages are matched with their counterpart integer pipeline stages for pipeline sequencing and synchronization (see Figure 8).

- *Prefetch.* The PF stage is the same as in the integer pipeline.
- *First decode.* The D1 stage is the same as in the integer pipeline.
- Second decode. The D2 stage is the same as in the integer pipeline.
- *Operand fetch.* In this E stage the FPU accesses both the data cache and the floating-point register file to fetch the operands necessary for the operation. When floating-point data is to be written to the data cache, the FPU converts internal data format into the appropriate memory representation. This stage matches the E stage of the integer pipeline.
- *First execute.* In the X1 stage the FPU executes the first steps of the floating-point computation. When floating-point data is read from the data cache, the FPU writes the incoming data into the floating-point register file.
- *Second execute.* In the X2 stage the FPU continues to execute the floating-point computation.
- *Write float.* In the WF stage the FPU completes the execution of the floating-point computation and writes the result into the floating-point register file.
- *Error reporting.* In the ER stage the FPU reports internal special situations that might require additional processing to complete execution and updates the floating-point status word.

The eight-stage pipeline in the FPU allows a single cycle throughput for most of the "basic" floating-point instructions such as floating-point add, subtract, multiply, and compare. This means that a sequence of basic floating-point instructions free from data dependencies would execute at a rate of one instruction per cycle, assuming instruction cache and data cache hits.

Data dependencies exist between floating-point instructions when a subsequent instruction uses the result of a preceding instruction. Since the actual computation of floating-point results takes place during X1, X2, and WF stages, special paths in the hardware allow other stages to be bypassed and present the result to the subsequent instruction upon generation. Consequently, the latency of the basic floating-point instructions is three cycles.

The X86 floating-point architecture supports single-precision (32-bit), double-precision (64-bit), and extended-precision (80-bit) floating-point operations. We chose to support all computation for the three precisions directly, by extending the data path width to support extended precision. Although this entailed using more devices for the implementation, it greatly simplified the microarchitecture while improving the performance. If smaller data paths were designed, special rerouting of the data within the FPU and several state machines or microcode sequencing would have been required for calculating the higher precision data.

Floating-point instructions execute in the U pipe and generally cannot be paired with any other integer or floating-point instructions (the one exception will be explained later). The design was tuned for instructions that use one 64-bit operand in memory with the other operand residing in the floating-point register file. Thus, these operations may execute at the maximum throughput rate, since a full stage (E stage) in the pipeline is dedicated to operand fetching. Although floating-point instructions use the U pipe during the E stage, the two ports to the data cache (which are used by the U pipe and the V pipe for integer operations) are used to bring 64-bit data to the FPU. Consequently, during intensive floating-point computation programs, the data cache access ports of the U pipe and V pipe operate concurrently with the floating-point computation. This behavior is similar to superscalar load-store RISC designs where load instructions execute in parallel with floating-point operations, and therefore deliver equivalent throughput of floating-point operations per cycle.

Microarchitecture overview. The floating-point unit of the Pentium microprocessor consists of six functional sections (see Figure 9).

The floating-point interface, register file, and control (FIRC) section is the only interface between the FPU and the rest of the CPU. Since the function of floating-point operations is usually self-contained within the floating-point computation core, concentrating all the interface logic in one section helped to create a modular design of the other sections. The FIRC section also contains most of the common floating-point resources: register file, centralized control logic, and safe instruction recognition logic (described later). FIRC can complete execution of instructions that do not need arithmetic compu-

tation. It dispatches the instructions requiring arithmetic computation to the arithmetic sections.

The floating-point exponent section (FEXP) calculates the exponent and the sign results for all the floating-point arithmetic operations. It interfaces with all the other arithmetic sections for all the necessary adjustments between the mantissa and the sign-and-exponent fields in the computation of floating-point results.

The floating-point multiplier section (FMUL) includes a full multiplier array to support single-precision (24-bit mantissa), double-precision (53-bit mantissa), and extended-precision (64-bit mantissa) multiplication and rounding within three cycles. FMUL executes all the floating-point multiplication operations. It is also used for integer multiplication, which is implemented through microcode control.

The floating-point adder section (FADD) executes all the "add" floating-point instructions, such as floating-point add, subtract, and compare. FADD also executes a large set of micro-operations that are used by microcode sequences in the calculation of complex instructions, such as binary coded decimal (BCD) operations, format conversions, and transcendental functions. The FADD section operates during the X1 and X2 stages of the floating-point pipeline and employs several wide adders and shifters to support high-speed arithmetic algorithms while maintaining maximum performance for all data precisions. The CPU achieves a latency of three cycles with a throughput of one cycle for all the operations directly executed by the FADD section for single-precision, double-precision, and extended-precision data.

The floating-point divider (FDIV) section executes the floating-point divide, remainder, and square-root instructions. It operates during the X1 and X2 pipeline stages and calculates two bits of the divide quotient every cycle. The overall instruction latency depends on the precision of the operation. FDIV uses its own sequencer for iterative computation during the X1 stage. The results are fully accurate in accordance with IEEE standard 754 and ready for rounding at the end of the X2 stage.

The floating-point rounder (FRND) section rounds the results delivered from the FADD and FDIV sections. It operates during the WF stage of the floating-point pipeline and delivers a rounded result according to the precision control and the rounding control, which are specified in the floating-point control word.

Safe instruction recognition. Floating-point computation requires longer execution times than integer computation. Pentium's floating-point pipeline uses eight stages, while the integer pipeline uses only five stages. Compatibility requires in-order instruction execution as well as precise exception reporting. To meet these requirements in the Pentium processor, floating-point instructions should not proceed beyond the X1 stage, that is, allow subsequent instructions to proceed beyond the E stage, unless the floating-point instruction is guaranteed to complete without causing an exception. Otherwise, an instruction may change the state of the CPU, while an earlier floating-point instruction (which has not yet completed) might cause an exception that requires a trap to a software exception handler.

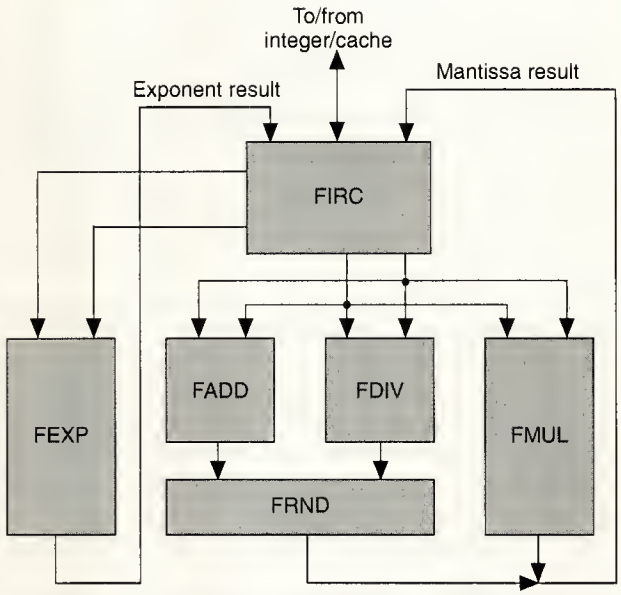


Figure 9. Floating-point unit block diagram.

To avoid a substantial performance loss due to stalling instructions until the exception status of a previous floating-point instruction is known, Pentium's floating-point unit employs a mechanism called safe instruction recognition (SIR). This logic determines whether a floating-point instruction is guaranteed to complete without creating an exception and therefore is considered "safe." If an instruction is safe, there is no need to stall the pipeline, and the maximum throughput can be obtained. If, however, the instruction is not safe, the pipeline stalls for three cycles until the unsafe instruction reaches the ER stage and a final determination of the exception status is made.

Six possible exceptions can occur on the Pentium microprocessor's floating-point operations: invalid operation, divide by zero, denormal operand, overflow, underflow, and inexact. The SIR logic needs to determine early in the floating pipeline—in the X1 stage—before any computation takes place whether the instruction is guaranteed to be exception free (safe) or not (unsafe). The first three of the six exceptions can be detected without any floating-point calculation. From the latter three exceptions, the inexact exception is usually "masked" by the operating system or the software application (using the precision mask, or PM, bit in the floating-point control word). Otherwise, a trap will occur whenever rounding of the result is necessary. When the pre-

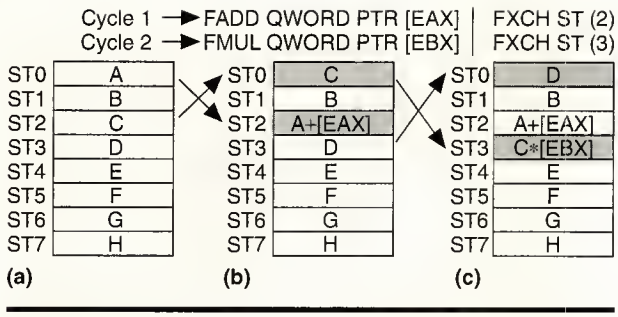


Figure 10. FXCH code example.

cision (inexact) exception is masked, the pipeline delivers the correctly rounded result directly. For overflow and underflow exceptions SIR logic uses an algorithm that monitors the exponent fields of the input operands to conclude the exception status (safe or unsafe).

In the X86 architecture the CPU stores floating-point operands in the floating-point register file with an extended-precision exponent, regardless of the precision control in the floating-point control word. The extended-precision exponent supports much greater range than the double-precision format. Overflow and underflow exceptions caused by converting the data into double-precision or single-precision formats occur only when storing the data into external memory. These characteristics of the X86 floating-point architecture give a unique advantage to the effectiveness of the SIR mechanism in the Pentium CPU, since the SIR algorithm can use the internal (extended-precision) exponent range. Thus, the occurrence of unsafe operations is extremely rare. Our evaluation of the SIR algorithm for the FPU design found no unsafe instructions in simulated execution of the SPEC89 floating-point benchmarks.

Register stack manipulation. The X86 floating-point instruction set uses the register file as a stack of eight registers in which the top of stack (TOS) acts as an accumulator of the results. Therefore, the top of the stack is used for the majority of the instructions as one of the source operands and, usually, as the destination register.

To improve the floating-point pipeline performance by optimizing the use of the floating-point register file, Pentium's FPU can execute the FXCH instruction in parallel with any basic floating-point operation. The FXCH instruction "swaps" the contents of the TOS register with another register in the floating-point register file. All the basic floating-point instructions may be paired with FXCH in the V pipe. The pair execute in parallel, even when data dependency between the two instructions in the pair exists. The use of parallel FXCH redirects the result of a floating-point operation to any selected register in the register file, while bringing a new operand to the top of the stack for immediate use by the next floating-point operation.

The example shown in Figure 10 illustrates the use of parallel FXCH. The code in the example generates the results of two independent floating-point calculations. The floating-point register file contains initial values prior to code execution: register ST0 (TOS) contains the value A, register ST1 contains value B, register ST2 contains value C, and so on. The two operations are

1) floating-point addition of value A with the 64-bit floating-point operand addressed by the general register EAX, and
2) floating-point multiplication of value C by the 64-bit floating-point operand addressed by the general register EBX.

When the floating-point pipeline is fully loaded and these two operations are part of the code sequence, the parallel FXCH allows the calculation to maintain the maximum throughput of one cycle per operation. Within one cycle the Pentium CPU writes the result of the addition to ST2, while the operand for the next operation moves to the top of the stack. On the next cycle, the processor writes the result of the multiplication to ST3, while the top of the stack contains value D, which may be used for a subsequent operation.

Transcendental instructions. The CPU supports all eight transcendental instructions that are defined in the instruction set through direct execution of microcode sequences. The transcendental instructions are

1) FSIN sine,
2) FCOS cosine,
3) FSINCOS sine and cosine,
4) FPTAN tangent,
5) FPATAN arctangent,
6) F2XM1 2**X – 1,
7) FYL2X Y * Log2(X), and
8) FYL2XP 1 Y * Log2(X+1)

We developed new, table-driven algorithms for the transcendental functions using polynomial approximation techniques. These algorithms substantially improved performance and accuracy over the i486 CPU implementation, which used the more traditional Cordic algorithms. The approximation tables reside in an on-chip ROM along with the other special constants that are used for floating-point computation.

The performance improvement of the transcendental instructions on the Pentium processor ranges from two to three times over the same instructions on the i486 CPU at the same frequency. The worst-case error for all the transcendental instructions is less than 1 ulp (unit in the last place) when rounding to nearest even and less than 1.5 ulps when rounding in other modes. The functions are guaranteed to be monotonic, with respect to the input operands, throughout the domain supported by the instruction.

Development process

Developing a highly integrated microprocessor involves collaboration between numerous teams having diverse technical specialties and working under the discipline of well-defined methodologies. A small team of architects and VLSI designers developed the initial concepts of the design. This group conducted feasibility studies of parallel instruction decoding and options for branch prediction techniques. Simultaneously, it evaluated performance by hand for short benchmarks and compiler optimizations. As initial directions were established, additional engineers participated, and subteams focused on the following areas:

1) behavioral modeling of the microarchitecture;
2) circuit feasibility design for caches, decoding PLAs (programmable logic arrays), floating-point data path, and other critical functions;
3) a flexible, trace-driven simulator of instruction timing for performance evaluation;
4) a prototype compiler; and
5) enhancements to existing instruction-tracing tools.

Throughout the design we refined the Pentium microprocessor using both top-down and bottom-up methods. Top-down refinement was accomplished through comprehensive characterization of executing benchmark work loads on the i486 CPU[4] and trace-driven experiments concerning alternative machine organizations conducted by architects using the performance simulator.

VLSI design engineers evaluating features critical to the targeted area and frequency refined the design from the bottom up. On two occasions in the design the accumulation of changes from bottom-up refinement caused the need for substantial restructuring of the microprocessor's global chip plan, or "die diets." On those occasions, interdisciplinary teams of specialists collaborated to brainstorm and evaluate ideas that could satisfy the global or local design constraints. In one instance, we found it necessary to refine the set of instructions that could be executed in parallel. Constraints had been assigned to the area and speed of the decoder PLAs. The VLSI designers identified combinations of instruction formats that would feasibly decode in parallel, and the compiler writers determined the optimal selection.

In the end, the measured performance of the Pentium microprocessor in production systems is within 2 percent of that predicted before the design was completed.

The logic validation of the Pentium processor design presented a major challenge to the design team. A comprehensive test base from the validation of previous X86 microprocessors was available. However, the Pentium processor microarchitecture introduced several new fundamental techniques, such as superscalar, write-back cache, and floating-point algorithms, that required a more rigorous verification methodology.

We used different validation approaches in pre-silicon testing of the Pentium microprocessor:

1) Architecture verification looked at the "black box" functionality from the programmer's point of view. We designed comprehensive tests to cover all possible aspects of the programming model and all the Pentium processor user-visible features.

Naming the Pentium processor

In naming the fifth generation of its compatible microprocessor line the Pentium processor, Intel departed from tradition. Pentium breaks a string of CPU products dating back to the late 1970s that used numerics (8086, 286, 386, 486).

"The natural course would be to call this chip the 586," said Andrew S. Grove, president and chief executive officer. "Unfortunately, we cannot trademark those numbers, which means that any company might call any chip a 586, even if it doesn't measure up to the real thing."

Pentium uses the Greek word for five, "pente," as its root to associate with the fifth-generation product and adds "-ium," a common ending from the periodic table of elements. Thus, the Pentium microprocessor is the fifth generation, a key element for future computing.

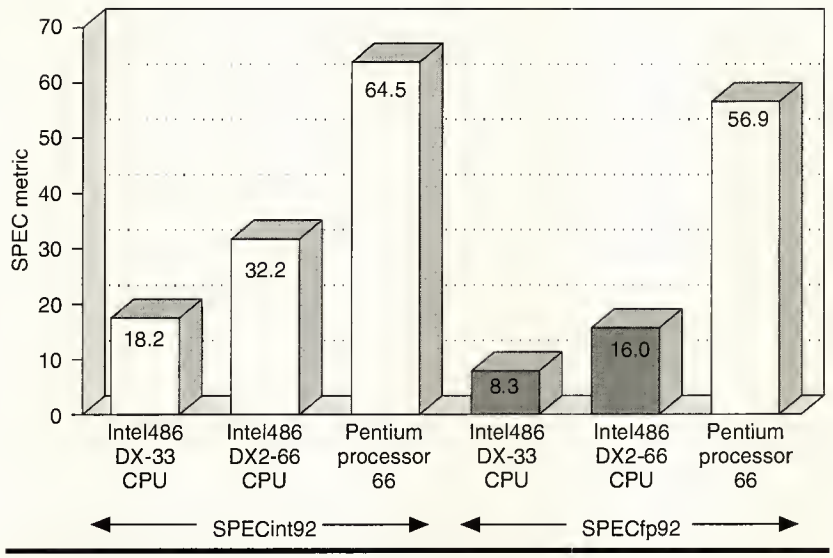


Figure 11. Pentium processor and i486 CPU performance for SPEC benchmarks.

2) Design verification checked the internal functionality from the point of view of a logic designer who would understand the behavior of every internal signal. This testing approach is considered a "white box" technique, in which tests are written to exercise all the internal logic and verify its correct behavior.
3) Random instruction testing was a valuable tool to cover all those situations that are rarely covered by the more traditional, handwritten tests. Running finely tuned random tests let us verify correct functionality by comparing the results generated by a logic design description of the Pentium processor to the results generated by a software-emulated model.
4) A logic-design hardware model (QuickTurn) enabled increased testing coverage capacity by allowing a much larger software base to run on the processor model before the first silicon was available. We ported the logic model of the Pentium processor onto a QuickTurn setup, which was capable of handling the complete design, and tested major operating systems and application programs before finalizing the design.

In addition to the general validation approach, we dedicated a special effort to verify the new algorithms employed by the FPU. We developed a high-level software simulator to evaluate the intricacies of the specific add, multiply, and divide algorithms used in the design. This simulator then evolved into a testing environment, allowing the verification of the FPU logic design model independently from the rest of the Pentium processor. Also, the new algorithms used for the floating-point transcendental functions required an extensive test strategy that verified the accuracy and monotonicity of the results throughout the development process, comparing the results to a "super accurate" software model. Eventually, when the first silicon of the Pentium processor was available for testing, we used automatic testing techniques to assure the correctness of the transcendental instructions.

Compiler optimizations

The compiler technology developed with the Pentium microprocessor includes machine-independent optimizations common to current high-performance compilers, such as inlining, unrolling, and other loop transformations. In addition, we used techniques specifically developed for the X86 architecture and tuned them for the Pentium processor's microarchitecture.

The X86 architecture has certain characteristics that require specialized optimization techniques different from those for RISC architectures. The architecture supports a variety of instruction formats for equivalent operations. Consequently, it is critical to select instruction formats that are decoded most efficiently by the processor. The X86 register set includes only eight integer and eight floating-point registers. We have found that common global register allocation techniques that assign variables to registers for the entire scope of a procedure are ineffective with such a limited number of registers. Registers must be allocated within a narrower scope and together with instruction scheduling.

The compiler schedules instructions to minimize interlocks and to maximize parallel execution for the Pentium processor's superscalar pipelines. These techniques also benefit performance on the i486 CPU (though to a lesser extent) because the processors' pipeline organizations are similar. The instruction-scheduling techniques have minimal impact on performance for the i386 CPU since that processor uses little pipelining. As explained in the description of the floating-point pipeline, the compiler schedules FXCH instructions to avoid floating-point register-stack dependencies.

THE PENTIUM MICROPROCESSOR employs superscalar integer pipelines, branch prediction, and a highly pipelined FPU to achieve the highest X86 performance levels available elsewhere while preserving binary compatibility with the X86 architecture. Figure 11 summarizes the performance of the Pentium microprocessor and the highest performance i486

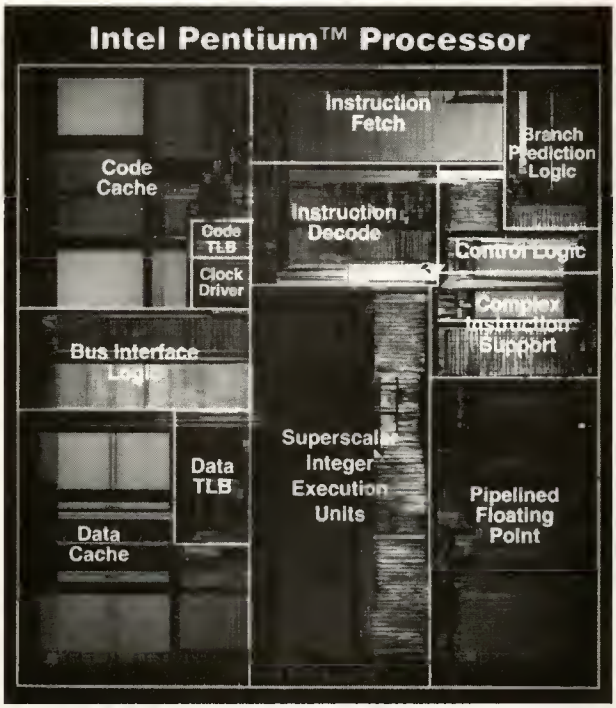


Figure 12. Die photograph.

CPU for the SPEC benchmarks in well-tuned systems. Figure 12 reproduces a photograph of the packaged circuit that integrates 3.1 million transistors.

Acknowledgments

The individuals who made substantial contributions to the Pentium processor's design are too numerous to list here, so instead we acknowledge groups of contributors. The VLSI design team applied their creativity and determined effort throughout the project. The compiler team developed and implemented novel optimization techniques. Software engineers in several groups developed instruction-tracing and performance simulation tools. Hardware engineers and technicians instrumented measurement and tracing systems. Architects facilitated and integrated efforts of these other teams. The efforts in architecture, optimizing compiler, and performance simulation involved collaboration between teams in Santa Clara and Israel.

References

1. *i486 Processor Programmer's Reference Manual*, Intel Corporation, Santa Clara, Calif., 1990.
2. *ANSI/IEEE Standard 754-1985 for Binary Floating-Point Arithmetic*, IEEE Computer Society Press, Los Alamitos, Calif., 1985.
3. John H. Crawford, "The i486 CPU: Executing Instructions in One Clock Cycle," *IEEE Micro*, Vol. 10, No. 1, Feb., 1990, pp. 27-36.
4. Tejpal Chadha and Partha Srinivasan, "The Intel386 CPU Family—Architecture & Performance Analysis," *Digest of Papers Compcon Spring 1992*, CS Press, Feb. 1992, pp. 332-337.
5. Robert F. Cmelik et al., "An Analysis of Mips and Sparc Instruction Set Utilization on the SPEC Benchmarks," *Proc. ASPLOS-IV Conf., Computer Architecture News*, Vol. 19, No. 2, Apr., 1991, pp. 290-302.

Donald Alpert is an architecture manager in Intel Corporation's Microprocessor Division. He holds responsibility for managing the architecture team that developed specifications and modeling and evaluating performance of the Pentium processor. Previously, he held various microprocessor development positions at National Semiconductor Corporation and Zilog.

Alpert received a BS degree from MIT and MS and PhD degrees from Stanford University, all in electrical engineering. He is a member of the IEEE Computer Society and the Association of Computing Machinery.

Dror Avnon is design manager of the floating-point unit of the Pentium processor. He holds responsibility for the microarchitecture, design, performance analysis, and verification for the FPU logic and microcode. He previously held design engineering positions at National Semiconductor Corporation, Computer Consoles, and Elscint.

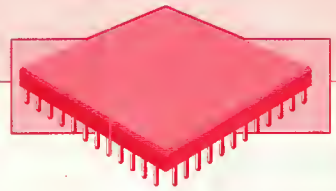
Avnon received a BSc degree in electronic engineering from Technion-Israel Institute of Technology in Haifa. He is a member of the IEEE Computer Society.

Direct questions to Dror Avnon, Intel Corporation, M/S RN2-27, 2200 Mission College Blvd., Santa Clara, CA 95052; davnon@mipos2.intel.com.

Reader Interest Survey

Indicate your interest in this article by circling the appropriate number on the Reader Service Card.

Low 159 Medium 160 High 161



Performance Features of the PA7100 Microprocessor

The PA7100 CPU is the first PA-RISC implementation to combine an integer core and floating-point coprocessor into a single-chip format. It also incorporates superscalar execution and supports clock rates of up to 100 MHz in standard 0.8-µm CMOS. Features such as a flexible primary cache organization and multiprocessing capability allow the device to scale into a variety of system applications, price ranges, and performance levels.

Tom Asprey

Gregory S. Averill

Eric DeLano

Russ Mason

Bill Weiner

Jeff Yetter

Hewlett-Packard

The PA7100 CPU is the seventh implementation of the Hewlett-Packard precision architecture, reduced instruction-set computer (PA-RISC) architecture. Since the first transistor-transistor logic PA-RISC was introduced in 1986, performance has doubled roughly every 18 months in subsequent implementations. As shown in Figure 1, the PA7100 has continued this performance growth trend with its introduction in 1992.

Time-to-market and binary compatibility with existing PA-RISC processors were important considerations in the design of the PA7100. Thus our design adapted much of the integer core, memory management circuitry, and cache interfaces from the earlier 66-MHz PA-RISC CPU.[1,2] See Figure 2 for a photograph of the PA7100 die. To extend the clock to a frequency of 100 MHz, we employed a combination of design evolution and improvements in CMOS IC technology. Measuring 1.42 × 1.42 cm, this chip uses the 0.8-µm CMOS26B process.

The performance goals of the PA7100 required that new features and circuits be developed in several key areas.[3,4] Most notable is a new floating-point coprocessor that is included on the chip and that delivers exceptional performance. Occupying less than 30 square millimeters of silicon area, the PA7100 coprocessor achieves an exceptionally low arithmetic latency. A new superscalar execution model allows the simultaneous dispatch of integer and floating-point instructions to further improve coprocessor performance. We improved the cache and translation look-aside buffer (TLB) subsystems, and also the interface to main memory. The PA7100 CPU retained hardware support for shared-memory multiprocessing. Our design adds a new two-way multiprocessing interconnection protocol that allows the CPU to scale into previously unavailable low-cost microprocessor applications.

Like all of the preceding PA-RISC implementations, the PA7100 relies on external static RAM arrays for primary cache. This arrangement allows system designers to configure data caches up to 2 Mbytes and instruction caches up to 1 Mbyte as well as scaled-down configurations in a wide range of price and performance levels.

While the primary design focus for the PA7100 was the high-performance technical desktop, the single-chip format lets us extend the architecture into the lowest price points yet available. The chip's unique set of features suits it equally well for a range of commercial applications. For example, a new type of hierarchical TLB known as a hardware table walker provides the best performance in virtual memory management of any PA-RISC to date.

To fully describe the PA7100 CPU, we first must discuss each of its major subsystems, emphasizing especially the design features and capabilities that support performance or scalability.

0272-1732/93/0600-0022$03.00 © 1993 IEEE

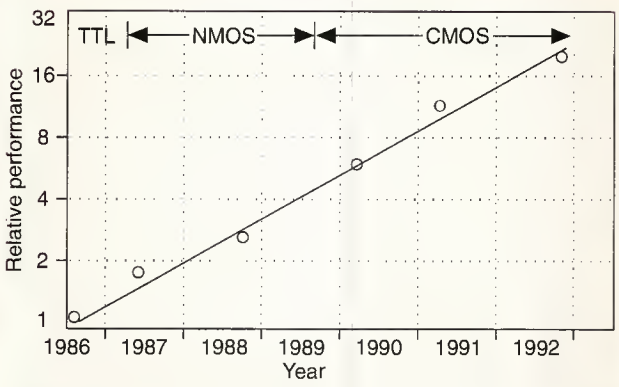


Figure 1. Quest for performance improvement.

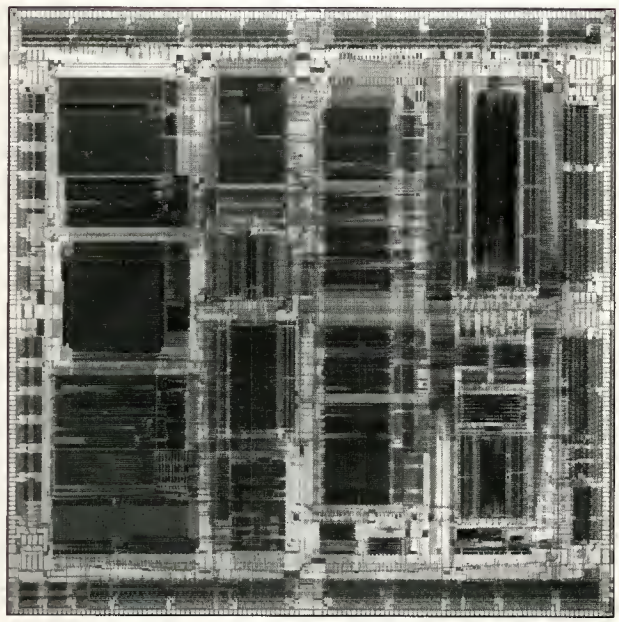


Figure 2. The PA7100.

Instruction execution

The instruction pipeline reflects many fundamental design choices. The most important of these choices is the cache structure. Although the use of on-chip rather than off-chip caches can facilitate faster pipeline clock rates, on-chip caches are usually not large enough to achieve acceptably low cache miss rates. For this reason, and also to provide for a high degree of price-to-performance scalability, the PA7100 employs off-chip caches made from industry-standard SRAMs. By dedicating three of the five pipeline cycles solely to cache access, we designed the PA7100 pipeline and system components so that only the cache SRAM read access time would limit the pipeline clock rate. The off-chip caches are cycled at the pipeline frequency of 100 MHz, and the chip can execute a load instruction every cycle. To maximize the processor frequency, the design relaxes write timing to two cycles so that this timing does not become critical. Single-cycle write operations would have reduced the processor frequency because of the time required to tri-state the SRAM outputs between reads and writes.

Figure 3 shows how our design executes various types of instructions in the pipeline. Each stage of the pipeline is divided into two equal phases, with the first three phases of the pipeline dedicated to instruction fetching. The next two phases include decode and data cache address generation. Next come three phases of data cache access (for loads and stores), and the last stage includes register write-back. The execution of integer operations, floating-point opera-

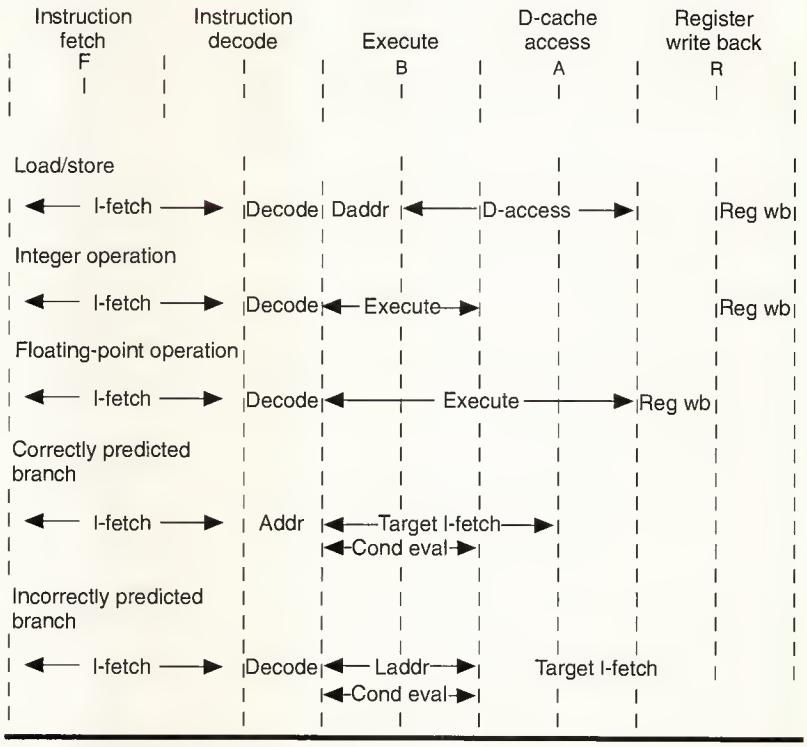


Figure 3. The instruction execution pipeline.

tions, and branches is straightforward. These operations are also shown in the figure.

Instruction execution pipeline

The partitioning of instruction execution into pipeline stages involves trade-offs between the clock rate and the number of pipeline interlock/branch penalties that are incurred by code. To increase the frequency of a pipeline, execution must be spread out over more stages, which increases the number of pipeline interlocks due to data/fetch dependencies. This helps explain why a 200-MHz computer does not always perform better than a 100-MHz computer. The PA7100 minimizes all pipeline interlock penalties subject to the constraint that the read cycle time of the cache RAMs determines the pipeline frequency. Table 1 shows some of the pipeline interlocks for the PA7100, the DEC Alpha processor, the Mips R4000, and the IBM RS/6000.[5,6] As expected, the PA7100 has fewer interlocks than the corresponding DEC Alpha. The effect of pipeline interlocks on overall performance depends on the benchmark, the compiler, and the memory system, issues that are beyond the scope of this discussion.

The PA7100 floating-point unit produces results with very low latencies. Most notably, the floating-point add/subtract, and multiply units have only two-cycle latency, and add/subtract/multiply instructions can be issued every cycle. A very high floating-point performance results, as does a simplified pipeline control design. In addition, the processor supports the superscalar execution of all floating-point operations with integer operations or integer and floating-point loads and stores. There are also no order or alignment constraints on the pair of instructions that are executed together.

Table 1. Pipeline interlocks for various microprocessors (penalties in cycles rather than instructions).

	PA7100	DEC Alpha	IBM RS/6000	Mips R4000
Maximum branch penalty	1	4	3	2
Maximum load use penalty	1	2/6*	1	2**
Maximum integer ALU interlock	0	1	0	0
Maximum floating-point ALU interlock	1	5	1	3
Maximum floating-point multiply interlock	1	5	1	6/7†
Maximum pipeline frequency (MHz)	100	200	62.5	100

* 2 for on-chip hit, 6 for off-chip hit
** 2 for on-chip hit, external cache operates at 25 MHz
† Single precision/double precision

The PA7100 implements a simple static branch prediction algorithm to allow for a zero-cycle branch penalty. This algorithm predicts that forward conditional branches are untaken and that backward branches are taken. Other current microprocessor designs have resorted to branch target caches and speculative execution to minimize their branch penalties, measures that are warranted only when the maximum branch penalty is high.[7] The PA-RISC architecture also provides inherent parallelism for conditional branches.[8] The conditional branches in the PA-RISC architecture perform operations such as compare, add, and move in parallel with the branch target calculation, and the branch condition is based on this operation.

The PA7100 implements a one-entry store buffer to minimize the store penalty. The chip writes the store buffer to cache at the same time that it performs the read tag operation for the next store instruction. This implementation has a maximum store penalty of one cycle. We can avoid this penalty simply by scheduling a non-load/store instruction immediately after the store instruction. Also note that because integer and floating-point results are calculated so quickly almost no benefit would come from implementing out-of-order stores or branches that wait for previous results.

Executing instructions in the PA7100 then is relatively simple and straightforward. The chip has to contend with very few pipeline interlocks. Nor does it require complex techniques to achieve a small number of average cycles per instruction. Because the pipeline interlocks in the PA7100 are so rare, the cycles lost to cache misses and TLB misses constitute the largest portion of the average number of cycles per instruction beyond the baseline cycles due to the pipeline. To minimize these cycles, the PA7100 has implemented an assortment of techniques and features that we will soon describe.

Caches

A major contribution to the performance of the PA7100 came from increasing the processor frequency to 100 MHz. Although we leveraged much of the cache design directly from the previous 66-MHz PA-RISC implementation used in our original Series 700 products, the PA7100 cache design required extensive changes. Reaching the required performance levels demanded more than an improved CMOS process and faster SRAMs.

Because the PA7100's cache load access time determines the maximum pipeline speed, the cache needed a careful

design approach. Not only did we need to allow for a high processor frequency but we also had to supply the number of data words and instructions per cycle that a very high performance RISC processor like the PA7100 requires. Reading two instructions per cycle allowed for superscalar execution, while writing two instructions at a time reduced penalties for I-cache misses. Reading and writing two data words at a time made double-word loads and stores possible, maximizing transfer rates between the CPU's registers and the data cache.

In addition to the speed and performance concerns already discussed, cost and scalability were also major considerations. The PA7100 was intended for use in systems that ranged from low-cost desktop computers to high-performance compute servers, super minicomputers, and large parallel arrays for supercomputers. Each of these applications has its own requirements for performance and cost. Low-end systems usually emphasize cost by reducing the size and speed of their caches. For high-end system designs, where performance is the dominating factor, a premium for larger and faster caches with higher performance is usually acceptable.

The PA7100 design addressed these issues by implementing its cache memory with separate external instruction and data caches connected to the CPU in a Harvard configuration. As shown in Figure 4, each cache connects to the CPU via its own independent 64-bit data path. The CPU then can read two data words and two instructions every cycle or write them every two cycles. At 100 MHz, this gives each bus a read bandwidth of 800 Mbytes/s and a write bandwidth of 400 Mbytes/s, giving the PA7100 cache a higher level of performance than most other RISC processors' primary "on-chip" caches.

The instruction and data caches are both virtually addressed. Data and instructions transfer to and from memory in eight-word cache lines. Both caches are directly mapped and scalable. The size of the instruction cache can be configured from 4 Kbytes to 1 Mbyte, while the data cache has a range of 4 Kbytes to 2 Mbytes. Using SRAMs widely available today, these primary external caches are vastly larger than the primary caches that can be reasonably fabricated on a single-chip CPU using current technologies. Table 2 lists the various cache configuration options and SRAM requirements. External caches also do not waste precious CPU real estate. Also, because our design uses industry-standard asynchronous SRAMs, memory can be upgraded at the lowest cost possible with newer memory technology as it becomes available. Keeping the primary caches off chip also allows for a range of configurations to provide for products at varying levels of price and performance.

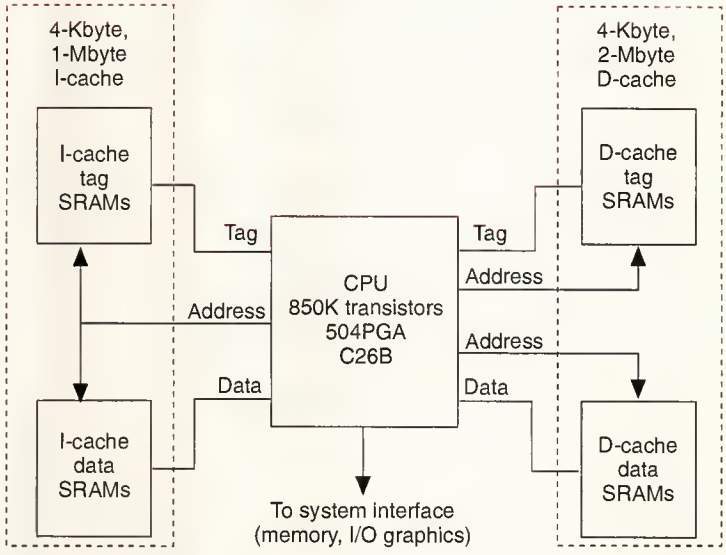


Figure 4. Processor block diagram.

Table 2. Cache configuration options and SRAM requirements.

Cache size (Kbytes)	Frequency (MHz)	SRAM size (Kbytes)	SRAM speed (ns)
256	50	32	15
256	66	32	12
1,024	96	128	9
256	99	32	9
256	132	32	7

Electrical design considerations. Although the external caches used by the PA7100 clearly have many advantages in performance, flexibility, CPU floor-space savings, and potential for future upgrades, their use did require a careful electrical design. Driving signals between chips at 100 MHz through printed circuit board traces and the chip's own package parasitic inductance and capacitance could have easily caused unacceptable delays. If we had not minimized these delays, the design would have required faster cache SRAMs or a reduced processor frequency, degrading the chip's performance. Requiring a faster SRAM would have quickly driven the cost of the cache too high, assuming an SRAM of the required speed was available at any price.

Complicating the design problem further was the high pin

Timing control had to be flexible to support a variety of specifications for different types of SRAMs, yet still make the fullest use of the SRAM for maximum processor frequency in a variety of configurations.

count required for the large buses between the PA7100 and its caches. Larger pin counts can increase the CPU package's footprint, which has the unfortunate effect of increasing the parasitic inductance and capacitance of the package. Besides increasing delays, high pin counts can also cause skews between the cache signals when some signals are affected more than others. Variations in the length of their interconnections and package load effects can generate such differences on the signals. Skew, like delay, can also lower the processor's frequency because it can limit the ability of a design to compensate for delay without decreasing the clock frequency.

To avoid the potential limitations these challenges might impose, the PA7100 design dealt with these issues directly. We tuned the PA7100 design to optimize the cache memory paths to get the highest processor frequency for a given speed of cache SRAM.

We decided to combine the floating-point unit with the integer processor unit into a single CPU chip for a variety of reasons, only some having to do with cache issues. Nevertheless, the cache design benefited because some of the signal pins that had previously been used for processor-to-coprocessor communication in the last PA-RISC implementation could now be used for CPU/cache signals instead. Even so, the CPU pin count increased by nearly 100 pins over its predecessor. We designed a new interstitial pin-grid array (PGA) package that not only accommodated the added pin count but even had a smaller footprint than PGA packages used by earlier PA-RISC designs. This actually improved the electrical performance of the package.

A single-chip processor configuration also cut the capacitive load on the instruction and data buses by more than half. Because these signals could now run directly between the CPU and the cache SRAMs and did not need routing to a third chip, the cache and CPU could fit much more closely together. This had several advantages. The design could match the signal lengths more evenly and keep them very short, eliminating the need for 189 electrical terminations and further minimizing delays and skews. If the signal loading had not been reduced and the terminations had been required to keep the electrical characteristics acceptable, routing of signal traces would have been longer and more complex. This would not only have increased delays and skews but also would have wasted a large amount of power and board space. Our design also saved the cost of the components for 189 signal terminations.

Another major part of the design was controlling the timing of signal transitions. The timing control had to be flexible to support a variety of specifications for different types of SRAMs, yet still make the fullest use of the SRAM for maximum processor frequency in a variety of configurations. The signals needed to transition early enough that the receiving chip, either the CPU or SRAM, had adequate time to act on the signal but not early enough to interrupt the chip's previous action or miss the last signal value. To control this timing, we used special clocks to signal at what time the driver circuits could force a transition or, in the case of a receiver circuit, could latch the signal value. These clocks also specified the point in time that either the CPU or the SRAM could drive its data lines. This arrangement avoids the situation where both the CPU and the SRAM attempt to drive the same line at the same time but to different voltage levels. We created the special clocks by using circuits that drive the CPU's internal clocks through printed circuit board traces to get delayed clocks. By varying the trace lengths, we could change the delays. We could then adjust the cache memory circuitry's timing for a variety of SRAM requirements without making changes to the CPU.

Instruction cache performance features. So that the PA7100 can issue and execute two instructions per cycle, not only must instructions be supplied at the same rate from cache but they must be supplied early enough for issuing to either the floating-point or integer unit without limiting the processor frequency. Early design efforts made it clear that the decode time could limit the processor frequency or require faster SRAMs. To keep the SRAM requirements as relaxed as possible, the design included dedicated predecoded bits for the instruction cache. This reduced the amount of decode required during instruction execution by effectively moving it to the time when the instructions are copied into the instruction cache. Because the predecode bits can directly steer the instructions to the proper execution unit, the design requires no special ordering for dual execution of two instructions to occur.

We also optimized the PA7100 to recover some of the time required to copy a cache line from memory into the instruction cache. By optimizing the execution of instructions, we allowed the processor to begin executing an instruction as soon as the instruction is returned from memory and before it is copied into cache. The PA7100 will continue to execute

instructions as they come from memory until the processor branches or its pipeline interlocks. Overlapping instruction execution and the manner of copy-in saves a maximum of six cycles per instruction-cache miss.

Data cache performance features. The PA7100 design improved the data cache's performance for both loads and stores to cache. Earlier PA-RISC processor implementations performed stores to the data cache in a three-cycle process. They first read the old tag and data, then verified a hit in the cache, while merging in the new data, and then finally wrote the merged data back to the cache. Stores now take two cycles. The store's tag read first overlaps with the previous store operation. Then it is used to verify that the line targeted by the store is a hit, or present in the cache, before the write begins. See Figure 5.

Overlapping the tag read of one address while data is written to a different address required separate address buses for the cache tag and data SRAMs. The dirty bit—usually associated with the tag and used to indicate whether or not a line is dirty and needs to be copied back to memory—remained with the data access. The dirty bit has to be marked for copy back to memory with the data store, and the tag address has already changed for the next load or store operation by the time the write begins, making this arrangement necessary.

Completing the store in two cycles also meant that we had to eliminate the data read/merge portion of a three-cycle read/merge/write store. For single-word stores, we used separate write controls to the cache data SRAMs for each of the two data words. This way only the data to be changed is written, and it need not be merged with the rest of the double word written to the data cache. To limit complexity, this was done only for word stores, as they are the great majority of stores. Much rarer byte and half-word stores still must be merged in the CPU and suffer a pipeline penalty.

To further enhance the performance of store operations, the design included a new encoding of the store instruction. It provides a hint to the hardware that if the stored data's cache line is not already in the cache, copying the cache line in from memory is unnecessary. The software can use this feature for block operations such as block moves, zeroing large memory spaces, and block copies. In these instances, the whole line will be changed anyway and fetching the old data from memory would be a waste of time. In such a case, unless the line it displaces in cache is marked dirty, only writing the tag for the new line into cache is needed. If the displaced cache line is dirty, it is first copied out to memory after which the new tag is written. This feature not only greatly increases the processor's performance for this kind of code, but also decreases the traffic on the memory bus, thus improving the performance of multiprocessor systems.

Two new PA7100 features helped to improve processor performance during a data cache miss. A data cache miss arises when a piece of data is not found in the cache and must have its corresponding cache line retrieved from memory. The "hit under miss" feature allows execution to continue even after a load or store has missed in the cache. Execution can continue until another cache miss occurs or the data from memory is required to complete execution of another instruction. Load-and-store operations can execute to lines in cache without stalling the CPU. During a store miss, word or double-word stores to the same cache line can execute. These options give software a great deal of flexibility in scheduling events for minimizing the penalty of a cache miss.

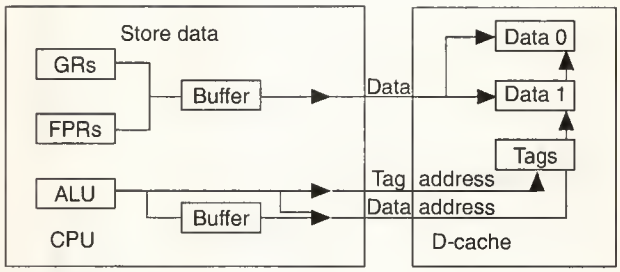


Figure 5. Store data path.

The design also included data cache "streaming" to allow execution to proceed as soon as possible. When the operand of an instruction is the target of an earlier load that missed in the cache, this feature allows the instruction to execute as soon as the critical word is returned from memory without waiting for the miss to complete.

The final data cache performance improvement involved the load-and-clear semaphore operation.[9] Under certain circumstances the operation can complete in the processor's cache, reducing its pipeline penalty to that of any store instruction and reducing traffic on the memory bus.

Virtual address translation (TLB)

We have optimized the process of translating the PA7100's 48-bit virtual addresses to real addresses in hardware to support translations handled by both hardware and software. The translation look-aside buffer, TLB, is a fully associative first-level hardware unified TLB (UTLB). It has 120 fixed 4-Kbyte page entries and 16 variable size entries, each of which can map 512-Kbyte to 64-Mbyte spaces. Any entry can be "locked in" by software to keep the entry from being replaced by translations for different virtual pages. To avoid conflicts between instruction and data accesses in the UTLB, the design includes a single-entry look-aside buffer for instruction translations. Each TLB entry contains a 36-bit virtual page number and its comparator, a 20-bit real page number (RPN), a 22-bit protection vector, and a valid bit. The data TLB entries also contain three debug and trap enable bits.

In addition to the hardware TLB, software maintains a vari-

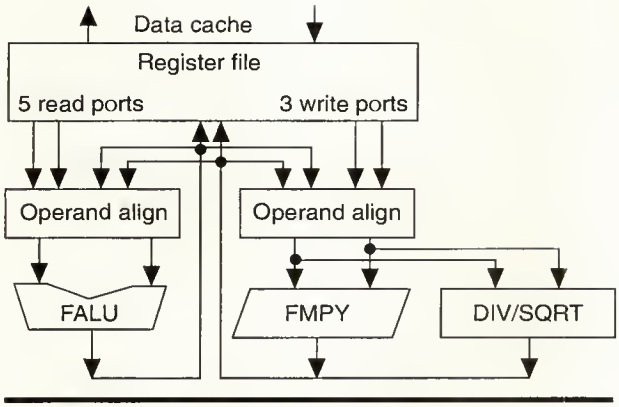


Figure 6. Floating-point block diagram.

Table 3. Floating-point instruction timing.

	Latency/dispatch (cycles)	
	Single precision	Double precision
ALU	2/1	2/1
Multiply	2/1	2/1
Multiply/ALU	2/1	2/1
Divide	8/8	15/15
Square root	8/8	15/15

able size, second-level TLB in memory that is read by a *hardware table walker* when there is a first-level miss in the hardware TLB. The physical page directory or PDIR contains this table.

We designed the hardware table walker (TLB miss handler) to reduce the TLB miss penalty, while keeping the PA7100 chip's area and complexity low. Encountering a first-level TLB miss, the PA7100 calculates the entry's address in the PDIR from the space register value and the virtual page number that missed in the UTLB. With the address determined, checking the PDIR entry for a valid matching tag begins. If there is a match, the chip inserts the real page number and protection information from the PDIR into the hardware UTLB and retranslates the original access. If the PDIR entry is not valid or does not match, its address passes to software and a trap is taken to the software TLB handler. Along with two other features this helps to minimize the software TLB miss penalty.

The general registers used in TLB software miss handling are automatically stored in "shadow" general registers as the trap is taken. The values are restored at the end of the trap handler with a special return-from-interruption instruction that restores the corresponding general registers from their shadow registers. New, fast TLB insertion instructions also help reduce the software handler's miss penalty. Using these optimizing features reduces TLB miss handling delays significantly. There is no penalty for a first-level hit in the UTLB, and the penalty for a miss is as little as ten cycles if the table entry is in cache.

By redesigning the TLB hardware, we took advantage of special opportunities to enhance TLB performance. Because large segments of memory can be mapped off and locked in the translation entry, the new design improves operating system and graphics software performance by keeping TLB misses low for large pieces of operating-system code, tables, and graphics frame buffers. The single-entry instruction look-aside buffer helps avoid contention with data accesses in the UTLB. Overlapping the buffer's update with the branch penalty also typically avoids its replacement penalty. The penalty for instruction TLB misses is, therefore, almost negligible when the entry is in the UTLB.

The PA7100 floating-point unit

The PA7100 floating-point unit contains five major subunits: floating-point pipeline control logic, a 32×64 register file, a floating-point arithmetic logic unit (FALU), a floating-point multiplier (FMPY), and a floating-point divide/square root unit (DIV/SQRT). See Figure 6. The floating-point data path implements IEEE 754 compliant single- and double-precision math. The floating-point unit provides exceptional floating-point performance for both technical and business computer systems. It achieves a peak execution rate of 200 Mflops at 100 MHz.

All floating-point operations except divide and square root (DIV/SQRT) are fully pipelined with a two-cycle latency for both single- and double-precision operands. The processor can issue an independent floating-point operation every cycle with no penalty cycles. Consecutive flops with a register dependency will incur a one-cycle penalty. Divides and square roots take 8 cycles in single-precision and 15 cycles in double-precision modes. Divides and square roots execute outside of the normal pipeline so that instruction execution does not stop until a dependency on the result register arises or another divide and square root is issued. This allows FALU and multiply instructions to execute in parallel with a divide or square root operation. Table 3 summarizes the timing for floating-point instructions.

Circuit density was a prime concern for the floating-point unit. Early in the design we saw that the highly parallelized algorithms commonly used in stand-alone coprocessor chip designs could not be compressed onto the CPU die. Furthermore, we realized that we would need fully combinatorial algorithms for the FALU and multiply circuits to achieve the

required level of performance. Using dynamic logic to exploit the well-known speed and density characteristics of that circuit type solved the problem. Typical dynamic circuits cannot perform inverted logic operations without introducing race hazards. To overcome this hurdle, we devised a system of self-timed logic. Using this method allowed us to design a multiplier and FALU that can compute full double-precision results in 20 ns.[10,11]

Floating-point register file. The floating-point register file contains thirty-two 64-bit floating-point registers. Registers 0-3 contain the status register and exception registers. Registers 4-31 serve as operands for the floating-point units. In addition, FR0 is hard-coded to floating point 0 when used as an operand. The floating-point instruction set can access each register as a 64-bit double word or as two 32-bit single words. The register file has three write ports and five read ports, which allows concurrent execution of a multiply, an add, and a load or store operation.

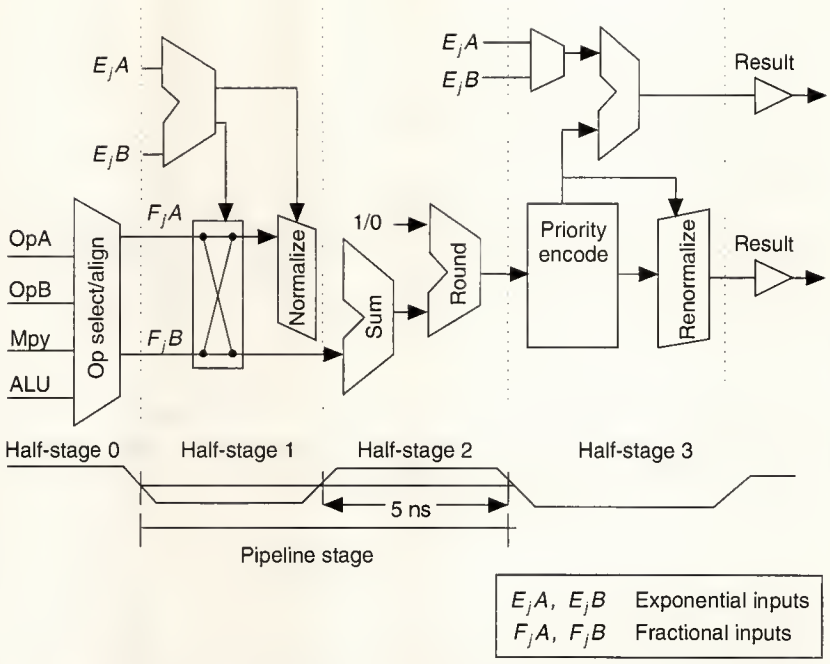


Figure 7. Floating-point ALU organization.

The floating-point instruction set includes instructions that perform more than one floating-point operation. Multiple-operation instructions are five-operand instructions that combine a three-operand multiply with a two-operand add or subtract. The format of the multiple-operation instructions is:

FMPYADD RM1, RM2, TM, RA, TA.

The operands specified by RM1 and RM2 are multiplied, and the result goes into the register specified by TM. The RA and TA fields specify the source operands for the FALU operation. The result of the FALU operation goes into the floating-point register specified by TA. The multiported register file allows for the simultaneous launch and completion of both the multiply and FALU operations.

The design provides extensive register bypass capability that reduces penalties for floating-point operations with register dependencies. We provided paths to bypass load data and floating-point operation results as operands to the floating-point units without first going through the register file. Also, a store bypass path bypasses floating-point operation results to the cache interface. A floating-point operation followed by a store of the target register incurs no penalty, even if these two instructions are simultaneously dispatched.

Floating-point ALU. The floating-point ALU performs add/subtract, compare/complement, and convert instructions for both single- and double-precision operands. The unit deviates from traditional implementations in that it performs floating-point additions, subtractions, and floating to/from integer conversions within a single functional unit. Traditional implementations perform additions and subtractions in one unit and conversions in another unit.

Shown in Figure 7, the floating-point ALU has four half-stages that correspond to the two states within the two-cycle latency. The first half stage, half-stage 0, latches the operands from the register file and checks for zero operands. Half-stage 1 shifts the significand of the smaller floating-point number to align the binary point. For a subtract operation, the smaller significand is complemented. Because the significands are unsigned, the design subtracts the smaller significand from the larger to avoid an extra complement operation in the case of a negative result. Half-stage 2 contains a 52-bit adder with rounding logic. Half-stage 3 contains a leading-one detector and a left shifter to postnormalize the result. Finally, the FALU drives the result back to the register file and optionally bypasses it to the operand buses or the cache store port.[2]

Integer to floating-point conversion involves normalizing the integer so that it consists of a significand in the form 1.xxxx with an appropriate exponent. This operation resembles the prenormalization step of addition or subtraction. The position of the most significant digit of the integer may be greater, however, than the number of bits in the destina-

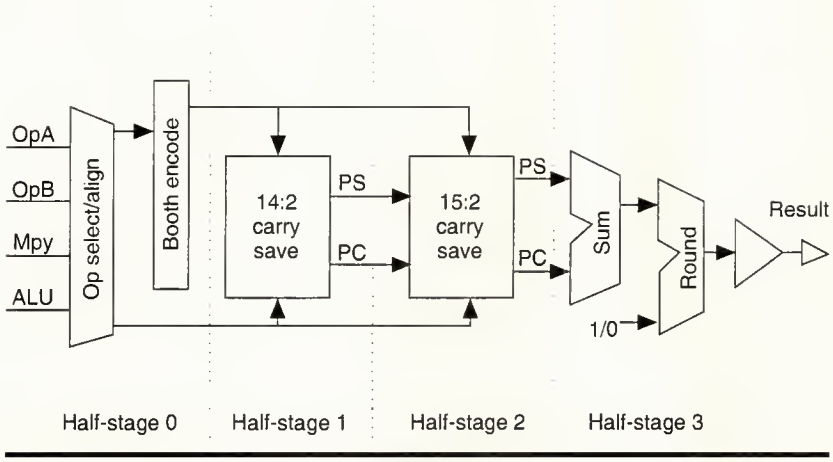


Figure 8. Floating-point multiplier organization.

tion significand. In this case, we must shift the integer to the right until it just fits into the most significant bit of the significand. Rounding is necessary if right shifting occurs. The additional hardware required beyond addition and subtraction is an integer leading-one detector to determine the amount of right shift.

Floating-point to integer conversions involve right shifting the significand of the floating-point number by the difference between the exponent and the exponent bias for the particular floating-point precision. Rounding becomes necessary if the right shift operation results in any lost digits. Having the generated integer in two's-complement form complicates the rounding process because the original significand is in sign-magnitude form. No additional hardware is required beyond that needed for addition.

Double-precision to single-precision floating-point conversion consists of right shifting the significand of the input operand 29 bit positions and rounding. Single-precision to double-precision floating-point conversion is a simple renormalization as is done for subtraction. Copy and absolute value operations require only multiplexers to add zero and modify the sign bit.

Because the hardware requirements for these operations are similar, the PA7100 floating-point ALU combines all of these operations into one functional unit. The additional hardware required beyond the traditional floating-point add/subtract unit is an integer leading-one detector, several multiplexers for the various operations, and additional control logic. This combined hardware approach saves significant area compared with traditional implementations.

Floating-point multiplier. The floating-point multiplier performs multiplies of single- and double-precision floating-point operands. In addition, integer multiplies of 32-bit unsigned integers provide a 64-bit result. Figure 8 diagrams the floating-point multiplier organization.

There are four half stages in the multiplier pipeline that correspond to the two states within the two-cycle latency. The first half stage, half-stage 0, provides both encoding of one of the significands and the start of the exponent add and rebias. Half-stage 1 completes the exponent add operations and performs half of the partial product summation (PS). Half-stage 2 completes the partial product summation. The carry-propagate addition to generate the significand product, rounding, and renormalization occur in half-stage 3. Finally, the floating-point multiplier drives the result back to the register file and optionally bypasses it to the operand buses or the cache store port.

The largest portion of the multiplier is the array that performs the generation and summation of partial products. Although the Wallace tree is generally accepted as the highest performance multiplier array structure, we made some concessions in the PA7100 array on behalf of silicon area constraints. The performance advantage is regained by the use of a high-speed dynamic full adder circuit that provides a summation delay as low as 350 ps.

IEEE rounding imposes one twist on the significand logic that mandated the development of a unique carry-propagate adder to obtain a compact, fast solution. For correct rounding, the design may need to increment the final significand. We designed a carry-select adder architecture in the multiplier rounding logic. This adder splits the word into delay-balanced sections. Within each section, the design implements two carry chains: one assuming the carry into the section is a one, and the other assuming the carry-in is zero. A second level of carry logic propagates the carry to the most significant bit position of the next section. This and other information determines the correct sum that can be rapidly produced by multiplexing the correct carry chain in each section to the single-gate-delay sum generator. In this way, the multiplier can generate the correct sum without duplicating the entire adder and multiplexing the correct answer. Also, the duplicate carry chain is part of the speed-enhancing multilevel carry scheme, not just overhead to generate the correct sum.

Floating-point divider. The divide/square root unit performs single- and double-precision operations. As diagrammed in Figure 9, we implemented this unit as a separate block that allows multiplies to continue even while a divide is in progress. The iterative divide/square root unit uses a modified radix-4 SRT algorithm that is a nonrestoring digit-by-digit method and is essentially an adaptation of hand division. (SRT division is a nonrestoring division algorithm—named for its developers, Sweeney, Robertson, and Tocher—that

uses a redundant-digit set.) Digit-by-digit division can be very slow, but by increasing the radix, each digit represents multiple bits of the quotient. The total number of iterations can then be reduced. As the radix increases, the complexity of SRT division grows exponentially.

Because of the simplicity of the radix-4 division hardware, the circuits can run at twice the system clock rate, achieving effective radix-16 performance with the low hardware cost of radix-4. The unit computes two bits of the quotient on each iteration. With two iterations during each clock cycle, four quotient bits are computed each clock cycle. With the divider running at twice the main clock frequency, the performance of the unit compares to high-performance Newton-Raphson dividers and requires only a fraction of the hardware cost.

Hardware underflow mode. The PA7100 implements a quick hardware underflow mode. This mode, which is enabled by setting a bit of the status register, eliminates the overhead of a trap handler when denormalized operands are present or when the result of an operation underflows. In the hardware underflow mode, operations that would normally signal the underflow exception return a zero result with no exception. In this mode, the PA7110 treats input denorms as signed zeroes. It detects the inexact flag and inexact exception just as in the IEEE mode except that it treats denormalized operands as signed zeroes. When a result is flushed to zero it sets the inexact flag. Note that when this mode is enabled, computations do not comply with the IEEE floating-point standard.

Memory and I/O interface

The PA7100 has a system interface bus (named P-bus) that services cache misses and I/O transactions. Although it is only 32-bits wide, the P-bus can operate at the pipeline clock frequency. In addition to the 32 address and data lines, the P-bus uses 17 protocol signals to allow data to flow on the P-bus at a rate near the bandwidth limit. The P-bus protocol includes cache-line-sized (32-byte) transactions, split read/return transactions, and single-cycle read requests. The P-bus also includes TLB and cache coherency transactions for supporting multiprocessor configurations. Because graphics applications are important for many systems that use the PA7100, we optimized the PA7100 pipeline and P-bus interface to sustain an I/O write bandwidth equal to one half of the P-bus bandwidth.

In a system design using the PA7100, a processor memory interface (PMI) connects the P-bus with the memory and I/O subsystems. Figure 10 illustrates a uniprocessor configuration. In workstation applications, we optimize the PMI to be a low-latency controller that is tightly coupled to the memory DRAM arrays and the I/O bus. In high-end commercial and technical server applications, the PMI is a bus converter that connects the P-bus to a higher-bandwidth system bus.

The earlier 66-MHz PA-RISC CPU used the P-bus as a system interface bus. We wanted to maintain compatibility with

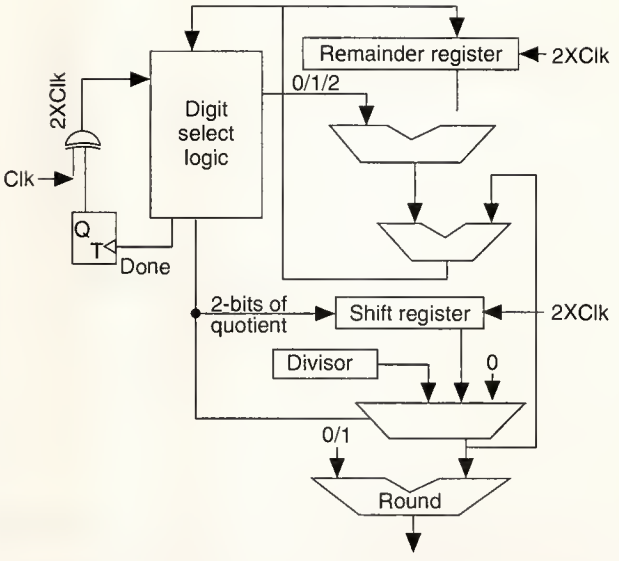


Figure 9. Floating-point divider unit.

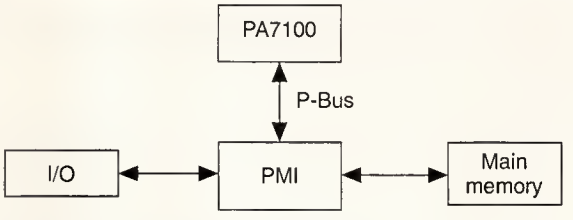


Figure 10. Uniprocessor configuration.

that system bus so that the 100-MHz PA7100 could serve as a processor upgrade for previous Hewlett-Packard computers that used a 66-MHz P-bus PMI. Buffering functionality, implemented in the P-bus interface, allows three pipeline-clock-to-P-bus-clock frequency ratios. These ratios are 1:1, 3:2, and 2:1. The 3:2 ratio allows the PA7100 to serve as a 99-MHz processor upgrade in the earlier 66-MHz systems. In addition to simple frequency conversion, the buffering control makes better use of the slower P-bus bandwidth by packing transactions more densely than would have occurred at the full-frequency, 1:1 ratio. The design accomplished this by eliminating wait states and by overlapping transactions.

Multiprocessing. The PA7100 includes hardware support for implementing shared-memory multiprocessor systems. The PA7100 implements the PA-RISC instructions for purging and flushing the caches and TLB.[9] The PA7100 supports two basic system configurations for constructing multiprocessor systems. One configuration is suitable for scalable, high-end systems, and the other makes possible a low-cost, dual-

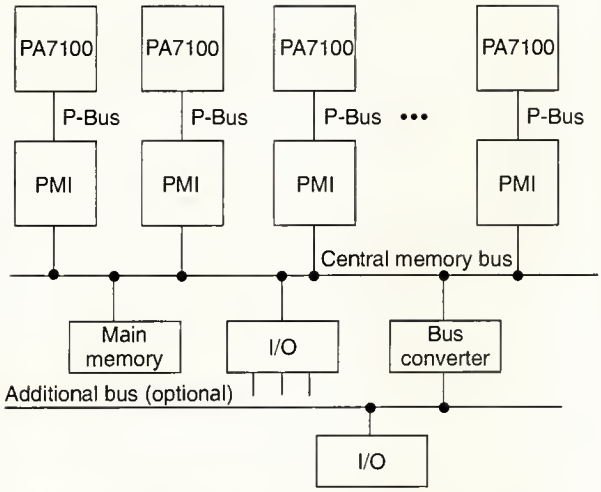


Figure 11. Scalable multiprocessor configuration.

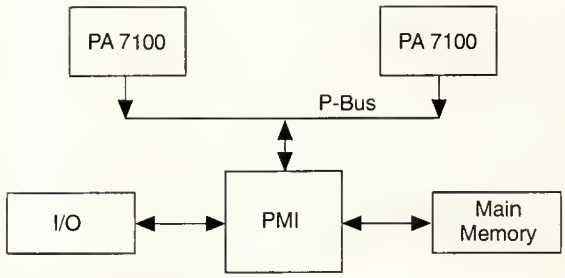


Figure 12. Dual-processor configuration.

processor system.

The PA7100 implements a write-back data cache policy. Each D-cache line can exist in one of four states:

1) invalid,
2) private-clean, meaning that the line is valid exclusively in the data cache of one processor and has not been modified with respect to the copy in main memory,
3) private-dirty, meaning that the line is valid exclusively in the data cache of one processor and has been modified but not yet posted to memory, and
4) shared, meaning that the line is unmodified and may be valid in more than one data cache.

The shared state is never used in a uniprocessor.

A coherent PA-RISC multiprocessor system using the PA7100 must behave as if there were logically a single data cache, a single instruction cache, and a single TLB. Cache flushes, cache purges, and TLB purges executed on one processor are broadcast to all other processors in the system. Hardware must maintain data cache coherency automatically. When a data-cache miss occurs, hardware will perform a cache coherence check by interrogating all other data caches in the system for the current data. The instruction cache is read-only, and instruction references need not be satisfied by a cache coherence check; software is responsible for modifications to the code stream.

Scalable high-end system. A scalable, high-end, shared-memory multiprocessor system using the PA7100 would be organized as shown in Figure 11. In this configuration, the PMI is responsible for maintaining cache and TLB coherency. Each PMI snoops on the shared bus, watching for coherent read, flush, and purge transactions issued by other modules on the bus. If a coherent transaction occurs, the PMI will initiate a coherency check or issue a flush or purge transaction to its CPU via the private P-bus connection. The PA7100 CPU will perform the requested cache or TLB action. If appropriate for the transaction type, the CPU then possibly will write back a private-dirty line over the P-bus to the PMI. When a dirty line is written back, the PMI will write the line on the shared bus so the line can be posted to memory, forwarded to another PMI-CPU pair that requested the line, or both. Sufficient information is available on the P-bus for the PMI to maintain a set of cache tags that mirrors the tags in the CPU cache. Coherent transactions therefore need to be forwarded to a CPU only when a target line is actually present in the cache.

This PMI-per-processor, shared-bus multiprocessor organization yields performance that scales well as the number of processors is increased. Implementing this organization is relatively expensive because the component count is high. Also, the bandwidth required to support multiple PA7100 processors requires an electrically sophisticated backplane and a highly interleaved memory controller design. This organization is appropriate for high-end server applications using the PA7100 in which the processing throughput that can be achieved justifies the cost of the implementation.

Low-cost dual-processor system. The PA7100 includes cache and TLB coherency functionality for implementing a low-cost, dual-processor system. Figure 12 illustrates the organization of this system. This organization does not require a PMI that implements the coherency functionality described for the scalable high-end system; the PA7100 processors carry all of the burden for maintaining coherency. The same low-latency PMI with tight coupling to DRAM and I/O that is used in a uniprocessor organization can be used for a dual-processor system. This organization supports the three pipeline-to-P-bus frequency ratios.

Essentially, the two processors share a single P-bus and appear to the PMI as a single, particularly bandwidth-hungry PA7100 CPU. Each CPU watches the transactions issued by

the other and internally performs the appropriate coherency transactions. When one processor issues a transaction that requires the other to write back a dirty line to memory, the first processor automatically, and transparently to the PMI, relinquishes control of the P-bus to the other. When one processor issues a read transaction for a line that is in the other processor's cache in the dirty-private state, the line is immediately transferred across the P-bus into the other cache. The line is not first posted to memory, and the requesting processor does not need to retry the read transaction.

Wanting to take advantage of the low-latency service the uniprocessor PMI can provide and desiring to maximize the use of the bandwidth available on the single P-bus, we paid particular attention to the arbitration protocol that the processors use when contending for the P-bus. The arbitration penalty is at most one state. If one processor wants to issue a transaction after having issued the previous transaction, and if the other processor is not simultaneously contending for the P-bus, the one processor may immediately issue its transaction without paying an additional arbitration penalty state. This feature allows a processor to issue a burst of transactions after using an arbitration state only for the first transaction. If one processor wants to issue a transaction after having not issued the previous transaction, it may do so after a one-state arbitration penalty. The arbitration protocol is fair in that each processor wishing to issue a P-bus transaction will wait for at most one of the other processor's transactions to complete before gaining control of the P-bus.

PA7100 methodologies

Bringing the PA7100 chip to fruition required a series of design, test, and verification methodologies. These methodologies allowed for a faster design cycle and greater quality of final product.

Design methodologies. Many of the choices made during the design phase of the PA7100 greatly influenced the performance and time to market of this processor. The PA7100 is composed of custom and semicustom block designs. To achieve maximum performance, we made critical circuits such as the TLB and IO structures from custom layouts. Since we leveraged many of the custom circuits on PA7100 from previous designs, our designers did not incur long development times. We made other circuits such as the data path and control logic from semicustom libraries that allowed for quick layout and easy design changes.

Our design places extra gates in and around major control blocks on the PA7100 CPU that could be used for the repair of functional and electrical defects. Using these spare gates for mending flaws meant that only metal layers needed to be changed, allowing for quicker turnaround of the silicon. Since some defects can mask other defects, decreasing the time it takes to evaluate bug fixes allows us to find other problems more quickly.

Since we leveraged many of the custom circuits on the PA7100 from previous designs, our designers did not incur long development times.

Test methodologies. To achieve the highest performance systems possible from the PA7100, testing for the part must be of the highest quality and allow for accurate binning. To attain this goal, the testing for the PA7100 employed a two-pronged approach: parallel-pin testing and serial-scan testing.

Serial testing, performed through a diagnostic port, allowed us to subdivide each PA7100 into blocks for thorough proof. The visibility provided by the large number of scan latches improved the process of finding processing defects. The scan latches also allowed for complete control of internal blocks, which permitted compact and effective vectors for finding faults.

Parallel-pin testing permitted an accurate determination of PA7100 speed and functional coverage of data blocks such as general registers, TLBs, and floating-point units. These blocks do not require complicated vector sequences for effective coverage. We generated parallel-pin vectors for the PA7100 from compiled code, allowing any code sequence found effective for stressing part speed to be used in the production test. This direct port from failing code sequences to test screen ensured the highest quality of speed binning.

Verification methodologies. We designed the verification techniques used for the PA7100 CPU, both before and after silicon, to bring out problems as quickly as possible. To ensure that first silicon would work, we used a description-level simulator with the descriptions employed at the lowest hierarchical level possible. We used schematic representations for all higher levels. This simulator could run code of any type, and we used an emulation program to check state-by-state results. Using this method meant that we were not restricted to self-checking code.

After silicon was produced, we checked the quality of design using many techniques. One new technique—using two types of pseudorandom code generators developed for the PA7100—proved very useful. We used the first type on the floating-point coprocessor. Floating-point emulation is a built-in function of Hewlett-Packard PA machines and is the check used by the random floating-point code generator. The code produced by this generator originated from a template that restricts the resulting code to certain instructions and sequences

Table 4. PA7100 workstation performance.

System	Frequency (MHz)	Cache (I/D) (Kbytes)	Memory (Mbytes)	SPECint 92	SPECfp 92	Dhrystone (MIPS)	SPECmark 89	SPECint 89	SPECfp 89	Linpack Dbl 100×100
HP9000/735	99	256/256	64	80.0	150.6	124	146.8	88.1	206.2	40.8
HP9000/715-50	50	64/64	32	36.5	72.1	62	69.0	40.2	99.0	13.2
HP9000/715-33	33	64/64	16	24.2	45.0	41	45.9	26.8	65.6	8.9

Table 5. PA7100 commercial system performance.

System	Frequency (MHz)	Cache (I/D) (Kbytes)	Memory (Mbytes)	TPC-A host-based	TPC-A client-server	AIM perf. index	AIM max. user load	AIM jobs/ minute	NHFS-stones
HP9000/G50	96	256/256	64	153	184	69.2	620	677	1,320 @39

of instructions. Its design involved comparing random code sequences that were executed against emulation of the same sequences. If the results differed, we saved the sequence for later debugging.

The second random code generator had a wider focus, and almost any instruction could be tested using this technique. We ran the template for this generator on a simulator and recorded the check sums. When the generator ran on the template, it would randomly change the environment under which the code was running. This was done by randomly inserting traps, cache misses, and modifying the mode of the PA7100 performance options. Once again, if a wrong check sum arose, we would log the failing code for later debugging. At the speed the PA7100 runs, random code can quickly squeeze out defects in the design.

Once we had discovered failing code sequences and determined that the cause of the failures was not readily reproducible in the simulators, we could debug the failure by using a scan path dumper. Many types of problems could not be reproduced in simulators, including speed paths, race conditions, or any electrical defect. We could compile any code and run it on the tester in the parallel-pins mode. Since the tester has full control over all chip pins, it can stop clocks at any time. With the clocks stopped at a predetermined state, the tester can interface with the PA7100's serial scan port and scan out the entire internal state of the chip. Doing this for consecutive states built up a log of the internal state of the part. We then compared this log with a log made by repeating the scan path dump at a passing frequency. If there were no passing points, we could compare the log to one produced from the description-level simulator. This technique prevented any problem from remaining misunderstood for very long.

THE PA7100 CPU IS CURRENTLY AVAILABLE in midrange commercial multi-user systems as well as a range of desktop workstations. Tables 4 and 5 give measured values for popular benchmarks. Hewlett-Packard is currently shipping these products in volume to its customers.

References

1. D. Tanksalvala et al., "A 90-MHz CMOS RISC CPU Designed for Sustained Performance," *Digest of Technical Papers, IEEE Solid-State Circuits Conf.*, Vol. 33, Piscataway, NJ, 1990, pp. 52-53.
2. C. Gleason et al., "CMOS Processor Circuit Design in Hewlett-Packard's Series 700 Workstations," *Proc. Int'l Conf. Computer Design*, IEEE Computer Society Press, Los Alamitos, Calif., 1991, pp. 288-292.
3. W. Jaffe et al., "A 200-Mflop PA-RISC Processor," *Proc. Hot Chips Symp. IV*, IEEE Computer Society Technical Committee on Microprocessors and Microcomputers, 1992, pp. 1.2.1-1.2.12.
4. E. DeLano et al., "A High Speed Superscalar PA-RISC Processor,"

Digest of Papers, Compcon, CS Press, 1992, pp. 116-121.
5. R.R. Oehler and M.W. Blasgen, "IBM RISC System/6000: Architecture and Performance," *IEEE Micro*, Vol. 11, No. 3, June 1991, pp. 14-17, 56-62.
6. S. Mirapuri, M. Woodcare, and N. Vasseghi, "The Mips R4000 Processor," *IEEE Micro*, Vol. 12, No. 2, Apr. 1992, pp. 10-22.
7. K. Diefendorf and M. Allen, "Organization of the Motorola 88110 Superscalar RISC Microprocessor," *IEEE Micro*, Vol. 12, No. 2, Apr. 1992, pp. 40-63.
8. R. Lee, "Pathlength Reduction Features in the PA-RISC Architecture," *Digest of Papers, Compcon*, CS Press, 1992.
9. R. Lee, "Precision Architecture," *Computer*, Vol. 22, No. 1, Jan. 1989, pp. 78-91.
10. C. Heikes et al., "The Design of 200-Mflop Floating Point Megacells for the PA7100," *Proc. Design Technology Conf.*, Hewlett-Packard, San Diego, Calif., May 1992, pp. 25-32.
11. J. Yetter, "A 100-MHz Superscalar PA-RISC CPU/Coprocessor Chip," *Digest of Technical Papers, Symp. VLSI Circuits*, 1992, pp. 12-13.

Tom Asprey is a member of the technical staff at Hewlett-Packard's Fort Collins site where he has been involved with CPU, cache controller, and system I/O chip designs for PA-RISC designs. He received a BS degree in electrical engineering from New Mexico State University. He is a member of the IEEE and the IEEE Computer Society.

Gregory S. Averill is a member of the technical staff at Hewlett-Packard where he has been involved in VLSI chip design and multiprocessor system design. He received a BS in electrical engineering from Brigham Young University and an MS in electrical engineering from Stanford University. He is a member of the IEEE and the IEEE Computer Society.

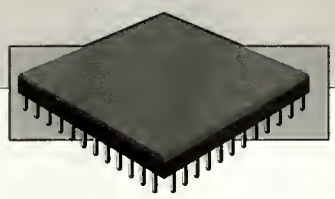
Eric DeLano is a member of the technical staff at Hewlett-Packard in Fort Collins, where he has been involved in the control design and definition of five VLSI CPU designs for the PA-RISC architecture. He received both BS and MEng degrees from the University of California at Berkeley.

Russ Mason, a member of the technical staff in Hewlett-Packard's Engineering Systems Lab, held responsibility for the design and verification of the PA7100 floating-point unit. He designed the floating-point register file and operand alignment circuits. In addition, Mason performed electrical verification of the PA7100 design. His current interests include the high performance circuit design for Hewlett-Packard's PA-RISC CPUs. He received a BSEE from Mississippi State University, Starkville, Mississippi.

Bill Weiner has participated in the design of high-performance PA-RISC computers as a member of the technical staff at Hewlett-Packard. He holds a BS in electrical engineering from Rensselaer Polytechnic Institute.

Jeff Yetter, as a member of the technical staff, has been engaged in the development and management of integrated RISC processors with Hewlett-Packard's System Technology Division. He holds a BS in computer engineering from the University of Illinois at Urbana.

Readers may direct questions concerning this article to Bill Weiner at Hewlett-Packard, 3404 E. Harmony Road, Fort Collins, CO 80525; or via e-mail at wnw@hpeswnw.fc.hp.com.



The Alpha AXP Architecture and 21064 Processor

The Alpha AXP 64-bit architecture forms the basis for a series of high-performance computer systems. Building on almost 10 years of internal research into reduced-instruction-set computer architecture, Alpha AXP emphasizes performance and longevity. The 21064 microprocessor is the first Alpha AXP implementation. Operating at speeds up to 200 MHz, this chip serves as the heart for current systems that offer the highest microprocessor-based performance in the industry.

Edward McLellan

Digital Equipment Corporation

The 64-bit Alpha AXP architecture[1,2] and the first implementation DECchip 21064 microprocessor grew out of a multiyear effort at Digital. Our aim was to develop a computer family capable of leadership performance for the foreseeable future over a wide variety of applications. Combining strengths in semiconductor technology, computer architecture, hardware design, operating systems, compilers, and applications software, this effort recently delivered a series of such machines. Systems range from personal computers to workstations to supercomputers. Operating systems support includes OpenVMS, full 64-bit Unix (DEC OSF/1), Microsoft Windows NT, and soon, native Novell NetWare. Figure 1 shows the packaged DECchip 21064 microprocessor.

Our rich history of computer design spans 35 years and includes the 16-bit PDP-11 and 32-bit VAX computer families. The Alpha AXP architecture represents a new step in that evolution, one that combines full 64-bit address and data capabilities with principles of RISC architecture. Roots of the AXP development go back to the mid 1980s, when multiple investigations of RISC technology culminated in the definition of the internal Prism architecture. That definition included the valuable experience of completely designing a 32-bit microprocessor.[3]

In 1988, a task force chartered with exploring future enhancements to the VAX concluded that a new architecture would soon be necessary to extend the increasingly cramped 32-bit addressing space of the VAX.[4] The Alpha AXP architecture went much further than that by addressing features such as multiple-instruction issue, multiple processors, and operating system independence. The Alpha AXP architecture benefits from the experience of a broad base of computer architects, hardware designers, and systems and applications software experts. The architecture strives to anticipate future trends as much as it attempts to provide current solutions. This architecture provides flexibility for both architectural evolution and hardware implementation over time in a variety of ways.

For any modern computer architecture to be successful, though, access to a strong semiconductor design and technology base is essential. Our semiconductor development began in the late 1970s with a double-metal NMOS process designed specifically to support high-performance microprocessors. Since then, our designers have developed four generations of CMOS technology. Each generation allows a straightforward path to shrink previous designs for advantages in speed, power, reliability, and cost. The 21064 microprocessor is designed in the CMOS-4 process, which offers 0.75-

0272-1732/93/0600-0036$03.00 © 1993 IEEE

μm feature sizes, three levels of aluminum interconnection, and a 3.3-volt power supply.

A new class of systems requires a tremendous amount of software development. Compatibility with older code was paramount for taking fullest advantage of the new architecture. The software task required more time than the hardware development. Therefore, we staged the hardware design to produce early development units that could assist software efforts. Prior to the final CMOS-4 version of the chip, we produced a CMOS-3 device that offered smaller caches and had no floating-point hardware support. This strategy allowed operating systems and internal developers to run code on actual AXP systems almost two years before the product shipment date. At the same time, the final hardware design got to take advantage of the latest semiconductor process advances.

Compatibility with a large, existing customer base of software also concerned us. Rather than burden the hardware with extensive support hooks, or restrict the architecture with compatibility issues, our design efforts adopted the idea of binary translation. Binary translation involves converting executable programs compiled for one hardware platform to another without requiring recompilation from original source code. For maximum reliability, the challenge also includes a runtime environment to support translation of almost all user mode applications and an interpreter to execute code that is not exposed by the initial translation. All of this must come together to produce a translated image that equals or exceeds the performance of the system being replaced. Translators for both VAX and Mips systems to AXP are available and have been invaluable in the highly successful migration of both system and user programs.

Alpha AXP architecture

Perhaps the most notable difference exhibited by the Alpha AXP is found not in a list of its features, but in its careful avoidance of quick-fix solutions to a variety of problems in computer design. Instead of a segmented address space, which can be more difficult to program, the design provides a large, 64-bit linear address space. Virtually all other computer manufacturers have 64-bit extensions planned, but only one other currently delivers 64-bit hardware. Only Alpha AXP offers a full 64-bit operating system with DEC OSF/1. A clean start rather than extension of a 32-bit architecture avoids hardware baggage that can include "orphan" 32-bit instructions (for example, 32-bit shifts) and other compatibility issues associated with old 32-bit software.

In Alpha AXP, all operations, including a small set for efficient 32-bit support, read and write full 64-bit quantities. To facilitate multiple-issue implementations, the architecture explicitly avoids condition codes, special registers, side effects, suppressed instructions, and branch delay slot instructions. These features fit well with single fetch and issue processors,

Figure 1. The packaged 21064 microprocessor.

but only complicate multiple-issue designs and often lead to performance bottlenecks. In a machine executing more than one instruction at a time, a single copy of any resource can become a point of contention. Likewise, a single skipped or forced instruction execution, as in the case of branch delay slots, does not fit well with the notion of a machine that fetches and executes multiple instructions each cycle.

The architecture also avoids direct hardware support for features that, although otherwise useful, are either uncommon or would likely limit the performance of anticipated systems through cycle-time restriction. Instead, the design provides support in a manner consistent with the architectural directions, but using software assistance for full functionality. Examples include the lack of direct-byte load/store instructions and precise arithmetic exceptions. A critical shift and multiplexer path is necessary for byte loads that can threaten cycle time. In addition, byte store operations require costly read-modify-write sequences in systems incorporating common error-correction code protection schemes. Byte writes with such ECC schemes can complicate and slow critical write-back cache designs. Recent experience has shown that some byte oriented codes run much faster when efficiently using the natural 64-bit (8-byte) data width and the byte manipulation support instructions provided.

Where byte operations are required, as in I/O support routines, designers of the first Alpha AXP PC have successfully used alternatives such as encoding sizes on address bits and encapsulating the byte manipulation code to port the Microsoft Windows NT operating system without changes to low-level driver code. In fact, byte manipulation encapsulation is identical to I/O operation encapsulation, which is necessary for using Intel X86 in and out instructions with high-level

The Alpha AXP architecture is a traditional RISC load-store architecture—all data moves between memory and registers without computation.

languages. A driver that already abstracts I/O operations need not be modified at all for use on Alpha AXP platforms.

As high-end processors such as Cray have done for years,[5] hardware support for arithmetic traps is imprecise with respect to the instruction stream. An operator can choose precise trap behavior when necessary through the use of the trap barrier instruction, typically during program debugging. General use of the trap barrier, however, can allow precise arithmetic exception behavior at all times without appreciably degrading performance. Measured differences on the 21064 range from less than 1 percent in integer to between 3 and 25 percent in floating-point codes. Advantages in cycle time and design complexity allowed by this approach, however, compare favorably with these differences.

The Alpha AXP architecture is a traditional RISC load-store architecture. That is, all data moves between memory and registers without computation. Computation is done between data in general-purpose registers only.

Operating system independence. Anticipating the need to support multiple operating system ports, a set of privileged software subroutines, called PALcode, can tailor some of the lowest level hardware-related tasks unique to a particular operating system. For flexibility in service, interruptions, exceptions, context switching, memory management, and error handling all have controlled entry points in PALcode. Neither the hardware nor the operating system then is burdened with a bad interface match, and the architecture itself is not biased toward a particular computing style. In addition, since PALcode mediates all access to physical hardware resources, including physical main memory and memory-mapped I/O device registers, users can also tailor the code for special purpose environments such as real-time and highly secure computing.

Addressing. Virtual addresses are a full 64-bits wide, although subsets are allowed. The AXP employs little-endian byte addressing, similar to Intel X86 and VAX computers. Systems can access both big- and little-endian data using the byte manipulation instructions with a single instruction modification to the sequence. In fact, Digital and its partners are building both big- and little-endian systems and software. Implementations may subset the address width, to a minimum of 43 bits with sign extension, but must check all 64 bits for compatibility with future systems. The AXP does virtual-to-physical-address mapping on a per-page basis, and its pages are 8 Kbytes with future expansion defined.

Data types. The fundamental unit of data is the 64-bit quadword, although the architecture also supports 32-bit longwords. Floating-point data types include both VAX and IEEE formats in both 32-bit single- and 64-bit double-precision formats. An extended-precision floating-point format is not included, but the designers have anticipated expansion by reserving a function field. Byte and word (16-bit) data types are not supported by direct load-and-store instructions but by short sequences of instructions. They can be manipulated in registers using normal arithmetic and the byte manipulation instructions.

Processor state. The hardware processor state includes separate 32-entry by 64-bit integer and floating-point register files. R31 is always zero in each file. Completing the required state are a longword-aligned, 64-bit program counter, floating-point control register for IEEE compliance, and a pair of lock registers for multiprocessor support. If the FETCH/FETCH_M instructions, or VAX-translated images are supported, additional hardware state is required.

The Privileged Architecture Library (PAL) gives designers the option of adding PAL state to the existing hardware state. The PALcode completes the architectural definition in an operating-system specific way. The hardware designers determine the implementation of PAL state, which can range from full hardware to full software or a combination of the two based on design constraints. Typical PALcode state include kernel stack pointer, user stack pointer, and translation look-aside buffers as well as a process-unique value for threads and a processor number for multiprocessor dispatch.

Instruction formats. As shown in Figure 2, the architecture uses four fundamental instruction formats: operate, memory, branch, and CALL_PAL. All instructions are 32-bits wide and contain zero to three register fields. To minimize register file port requirements, register B (RB) is never written and register C (RC) is never read.

The operate format includes arithmetic, logical, shift, and byte manipulation instructions. Scaled add/subtract and compare bytes instructions allow efficient operation on arrays and strings. Conditional move instructions for both integer and floating-point data, which test one input operand and optionally transfer data from another, remove branches in favor of a single instruction. Rather than using a single condition code location, the compare instructions write directly to any general-purpose register. They include an unsigned comparison operation for extended-precision arithmetic. There is no integer divide instruction. Where necessary, a 128-bit multiply can be used for emulation. The architecture enables traps on a per-instruction basis to avoid mode registers, and

provides some longword (32-bit) operations for compatibility.

Memory format instructions are mainly loads and stores, but also include some additional instructions. Loads and stores use two registers, specifying a base address and a data source or destination. The effective address calculation sign extends a 16-bit displacement to 64 bits and adds the 64-bit base register value. The architecture also provides load-and-store operations for longword (32-bit) quantities. General operations move aligned data quantities and trap on unaligned references, but instructions that mask the unaligned address bits and do not trap are available for use with the byte manipulation instructions. Calculated jump instructions also use the memory format; these instructions determine the target address directly from the base address without using the displacement field. The unused bits, however, are defined as hints for hardware prefetching mechanisms to improve pipeline efficiency. The additional hint information designates a likely target, allowing the hardware to continue fetching before the true target is available from the register file. If the hint is wrong, a misprediction restart costs no more time than if the hardware stalled waiting for the true address. A pair of load address instructions also use the memory format and allow a convenient way to create large constants using the 16-bit displacement field.

The design provides branch format instructions for both integer and floating-point data. These instructions test a single register for an operation-code-specified condition, and either branch to the target or fall through. To calculate targets, the instructions add a 21-bit longword displacement field to the updated program code (PC) resulting in a ±4 Mbyte relative branch range. The large range effectively reduces the need for branches around or to other branches.

The CALL_PAL format instruction contains only a 6-bit operation code field and 26-bit function field. There are no explicit registers because individual instructions can be redefined for specific use. When executed, these instructions dispatch to PAL routines that perform an atomic, or uninterruptable sequence of instructions. CALL_PAL instructions then can serve to emulate complex instruction-set computer functionality.

Shared-memory multiprocessing. Scalable performance was an integral part of the architectural definition. Since cycle time and multi-issues for single processors are likely to become limiting factors over the lifetime of the architecture, multiprocessor support was critical to achieving both performance and longevity goals. The basic multiprocessor interlocking primitive for updating a shared-memory location is a RISC-style load-locked, in-register modify, store-conditional sequence of instructions. If the sequence completes without interruption, exception, or an interfering write from another processor, the store-conditional instruction succeeds and returns status indicating that an atomic update was performed. Otherwise, the store-conditional fails, and the program must branch back and retry the sequence. This mechanism scales well with processor performance and allows multiple simultaneous noninterfering sequences.

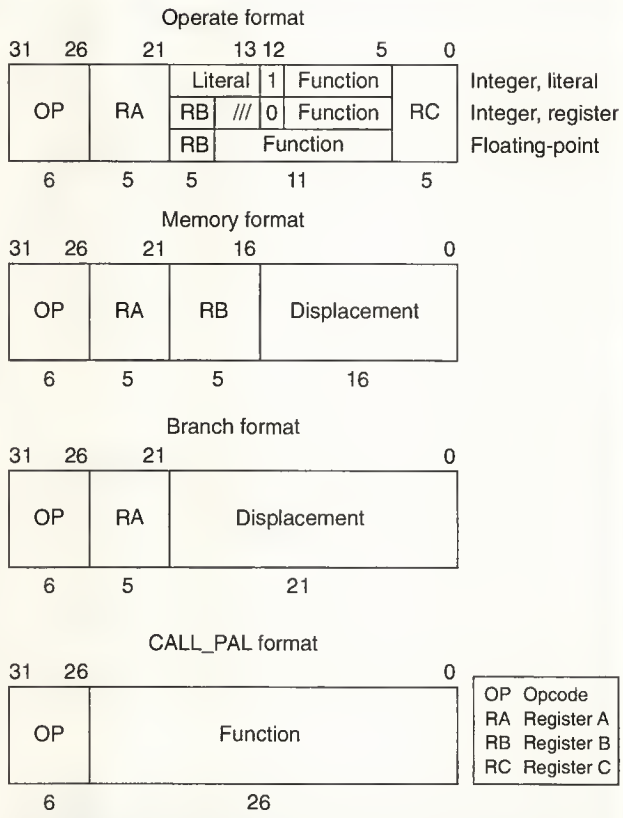


Figure 2. Instruction formats.

The Alpha AXP architecture is the first RISC architecture to offer a relaxed, or weak memory-ordering model. SPARC V9 and PowerPC have more recently announced support for a weak-ordering model. Relaxed ordering implies that the sequence of reads and writes as viewed by another processor need not be in order. Multiprocessors that employ strict-ordering models are possible, but can be subject to performance limitations. For example, if a processor is designed to retry writes that result in errors, a strict-ordering model implies that the retry must complete before any other read or write occurs. This constraint excludes pipelined memory systems that would otherwise allow operations begun prior to the error to complete before, and out of order with the retry.

When strict ordering is required, as is the case in some I/O or multiprocessor synchronization operations, the Alpha AXP architecture specifies a memory barrier (MB) instruction to force serialization of operations. Software then controls serialization, enforcing it only when necessary. The lack of implicit ordering enables a variety of high-performance implementation techniques.

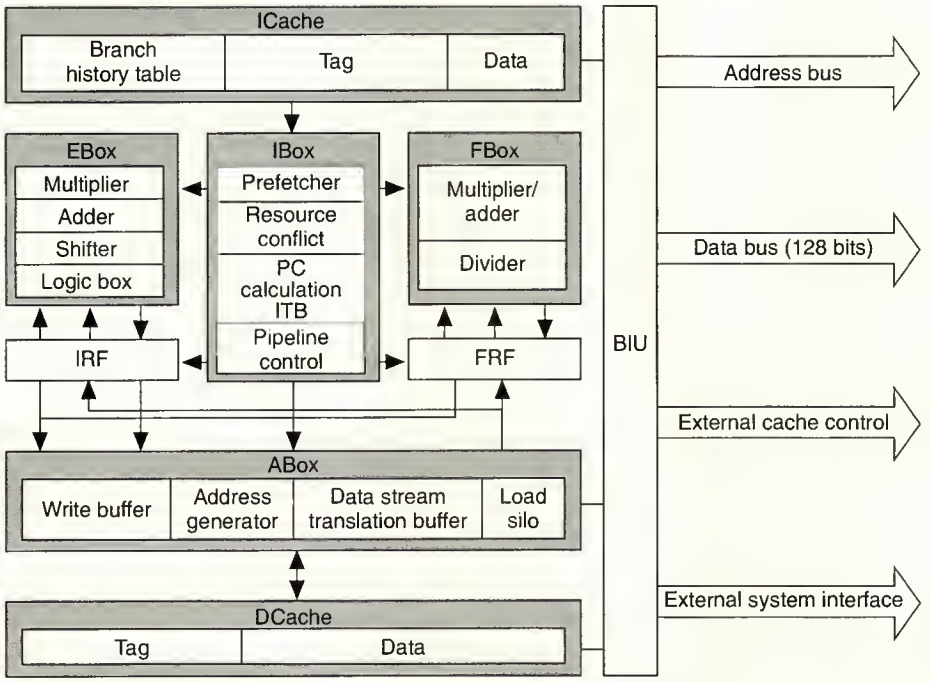


Figure 3. The 21064 block diagram.

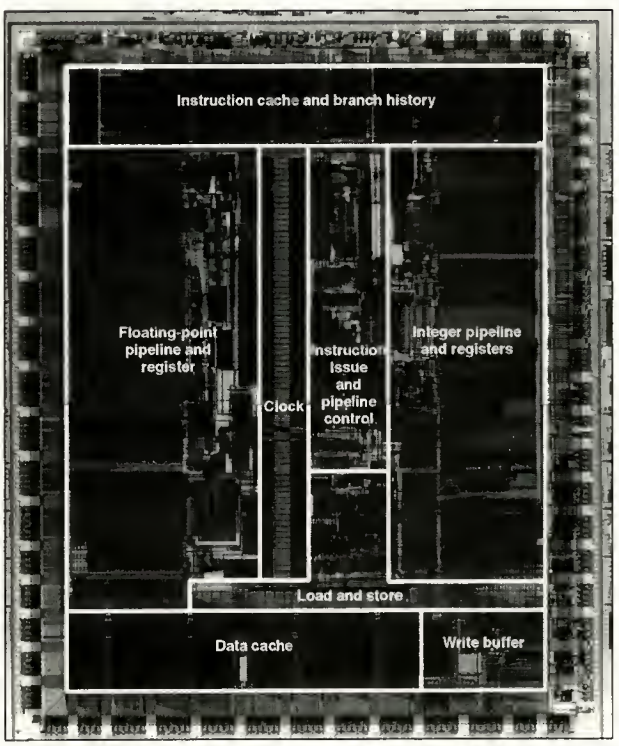


Figure 4. The 1.4 × 1.7-cm CMOS 21064 chip.

The 21064

The 21064 microprocessor is the first implementation of the Alpha AXP architecture.[6,7] This 1.4 × 1.7-cm CMOS chip incorporates 1.68 million transistors using a 0.75-μm, three-metal process. Figure 3 shows the chip's block diagram and Figure 4 shows a photograph of the chip itself. The design provides high performance through superscalar (two instruction issue) operation with an exceptionally high frequency internal clock cycle. Production chips and systems are available at clock speeds up to 200 MHz.[8] Despite the fast internal cycle time, the 21064 provides a flexible external interface that can easily accommodate a range of system designs. These designs are well within the range of standard interface devices due to the on-chip programmable system clock. System designs can run the CPU at from two to eight times the system clock frequency. Initial system designs range from PC to workstation to supercomputer class. They offer the highest microprocessor-based system performance in the industry as measured by the System Performance Evaluation Corporation (SPEC) suite of benchmark programs. (SPEC benchmarks, a series of programs measuring both speed and throughput, have become the standard for measuring computer performance.)

Cycle time implications. Overall performance involves many factors, but the two controlled primarily by the microprocessor designer are cycle time and the amount of work or instructions completed per cycle. Experience developing an earlier short cycle-time microprocessor[1] combined with simulations of possible design alternatives reinforced the RISC conclusions that the more potent lever was cycle time reduction. Based on the aggressive cycle time goal of typical parts at 150 MHz and fast parts at 200 MHz, the design supports two levels of cache hierarchy. The high speeds require on-chip caches to supply data and instructions at the cycle time rate (5 ns). However, die size and speed constraints limit the maximum size of that cache, which then can reduce performance. A large off-chip, second-level cache can mitigate this effect. The combination provides better overall performance and promises a greater rate of improvement as process density increases. The relative performance gain in increasing a small cache is greater than made available by increasing a large cache. In addition, a

small on-chip cache can scale with CPU cycle time much better than a large off-chip cache, allowing a design to take full advantage of advances in process technology.

To maintain the cycle time goals, we carefully evaluated all potential features—even a slight cycle time slip would likely cost more performance than the feature could give us. This philosophy extended through the ongoing architectural definition to avoid requirements that could limit implementation. For example, we postponed the decision regarding inclusion of the scaled add-and-subtract instructions in the architecture until we could demonstrate that implementations would not incur an adverse internal cycle time hit.

Dual issue. Dual-issue capabilities also exhibit cycle time influences. Rather than allow complete dual-issue flexibility, which only improves performance by an approximate increment of 2 percent over the final design, our design slightly restricts the instruction pairs for multiple issue. Compilers can group dual-issue operations in pairs when possible, but excess code expansion arises due to instruction padding if they are required to always align within the pair as well. When necessary, the hardware swaps pairs capable of dual issue. Hardware also serializes pairs that cannot dual issue to streamline internal control and data paths. All important pairs allowed by combinations of functional units can dual issue.

Since load-and-store operations predominate in RISC codes, the design provides a separate address unit to allow load and stores to execute with operate instructions. Table 1 shows the general instruction pairings for dual issue. There are only two exceptions to these rules. Branches cannot dual issue with stores of the same format because they share a register file port, and stores or branches cannot dual issue with operates of a different format because they share an instruction bus. For example, integer stores cannot dual issue with floating-point operates.

Table 1. General dual-issue rules.

Instruction A	Instruction B
Integer operate	Floating-point operate
Load/store	Operate
Branch	Load/store/operate

Pipeline. As shown in Figure 5, the integer and floating-point pipelines are, respectively, seven- and 10-stages deep. The first four stages are common to the two pipes and comprise the instruction fetch-and-issue section of the chip. Each stage can process up to two instructions in parallel. In the instruction fetch (IF) stage, the processor fetches a pair of instructions each cycle from the 8-Kbyte instruction cache. The swap (SW) stage controls instruction prefetching, doing branch prediction and cache index calculation as well as the swap or serialization operation described earlier. The issue-zero (I0) stage checks for intrafetch dependencies. This stage also completes the decoding and set up for the issue-one (I1) stage, which includes the register conflict detection and instruction issue to the datapath function units. Both the integer and floating-point register files are read in the I1 stage to supply data to integer, floating-point, load/store, and branch calculation units as shown.

The integer calculation pipeline writes results back to the integer register file in pipe stage 6, while floating-point calculations write to the floating-point register file in stage 9. Pipe stage 4 resolves branches, after which the prefetcher is redirected, resulting in a four-cycle misprediction penalty.

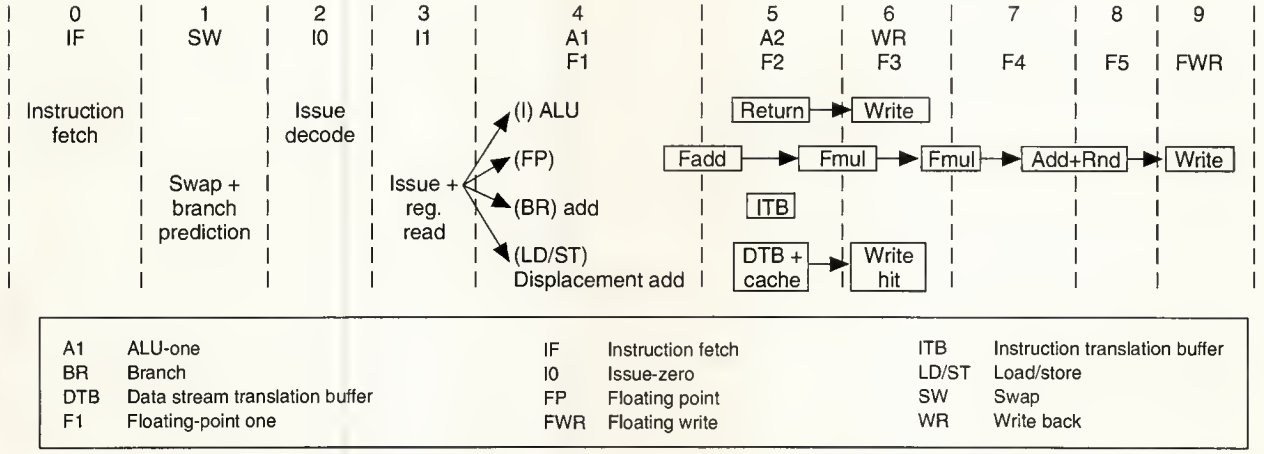


Figure 5. Pipeline.

Branch instructions and condition codes

Branches pose an increasingly severe problem in heavily pipelined and superscalar computer designs as a pipeline flush costs more potential instruction issue slots. The Alpha AXP architecture offers an advantage in handling branch instructions. Most computer architectures include condition codes to hold the result of arithmetic operations that can later be used to determine the outcome of branches. Unfortunately, the condition code register itself can become a point of resource contention if you assume that multiple instructions are executed simultaneously.

Recognizing this problem, some architectures offer combined compare-and-branch instructions that both reduce the number of instructions and eliminate the intermediate condition code storage. These instructions, though, force the arithmetic operation to be performed immediately before the branch. Arithmetic operations typically are one of the last pipeline stages, which therefore increases the misprediction penalty. In a superscalar implementation, it may be possible to resolve the branch decision early and overlap its execution with unrelated instructions.

The Alpha AXP architecture does not use condition codes. Instead, it resolves all branches based on the test of a single register. In effect, any register can hold branch condition information, eliminating the resource problem. In addition, since the branch need only test a single register, all that is required to resolve any branch immediately after reading the register file are the most- and least-significant bits, and a zero detection, which could be stored at register write time.

Loads that hit in the 8-Kbyte on-chip data cache write the associated integer or floating-point register file in pipe stage 6, simultaneously with integer calculations. The chip checks data cache misses in the large off-chip backup cache under complete CPU control. It only generates a system read block command if both caches miss. Instruction cache misses also get checked in the backup cache and fetch a second sequential 32-byte block into an on-chip streaming buffer. If the CPU reports an additional instruction cache miss that hits in the stream buffer, the data is simply moved into the cache and the next block is fetched in parallel with instruction execution.

Despite the apparent two-cycle delay for integer calculations, data is available after the first cycle in most cases. As shown in Figure 6, an extensive set of data bypass paths allows many back-to-back dependent operations to execute at fully pipelined speeds. In all, the chip uses 45 different bypass paths to minimize the effect of pipeline latency on dependent operations. All register conflict checking is done in hardware. Up to 22 operations thus can be in various stages of completion simultaneously, including 14 within pipeline stages 0 to 6, three in the extended floating-point pipe, three outstanding load misses, a floating-point division, and an integer multiplication.

Branch handling. With such a deep pipeline, branch handling is particularly important. The Alpha AXP architecture reduces branches through the use of the conditional move instructions and also includes hints for hardware-assisted branch prediction. The 21064 uses these hints and includes additional features. The chip can statically predict conditional branches using the sign of the displacement field to predict backward branches as taken and forward branches as not taken. In addition, the 21064 contains a 2K by 1-bit branch history table for dynamic prediction that provides approximately 80 percent accuracy for most programs.

The instruction prefetcher also contains a last-in, first-out stack of recent subroutine return addresses used to predict return paths for subroutines. The stack is repaired during pipeline flushes and therefore allows two additional benefits. As explained in the box, branch misprediction can become a performance issue at these fast cycle times due to the length of the pipeline. The chip uses the subroutine return stack to source the alternate, and correct, branch path one cycle earlier than the program counter datapath could provide it upon branch misprediction. In addition, the stack is used to accurately predict the return address for exceptions, since most exceptions (such as translation look-aside buffer miss) as measured by frequency, return to the original routine.

Integer unit. The integer register file contains thirty-two 64-bit general-purpose registers. It provides six ports, including four reads and two writes to allow the parallel execution of both integer calculations and load, store, or branch operations. The data path includes dedicated adder, shifter, multiplier, and logic units. Both the logic unit and adder provide results in one cycle. The shifter requires two cycles for results but is fully pipelined. The multiplier is not pipelined for area savings, and it supports the Alpha AXP UMULH instruction that returns the upper 64 bits of a 128-bit product for extended-precision operations and integer division support.

Floating-point unit. The floating-point unit combines maximum throughput with short latencies. It contains a 32-entry by 64-bit register file with three read and two write ports. The multiplier uses a radix-8 Booth algorithm in a fully pipelined two-way interleaved array. The rounding operation is completed simultaneously with the last adder stage for all operations. For compatibility, this unit supports both VAX and IEEE single- and double-precision data formats. It can initiate new instructions every cycle with dependent operations requiring six-cycle latency. The fast cycle-time goal trans-

lates into longer total latency as measured in cycles. For many floating-point codes, though, throughput and optimized compiler algorithms deliver exceptional performance as demonstrated by the >200 SPECfp92 values measured on DEC 10000 systems.

Address unit. The address unit performs all load-and-store operations. To do so in parallel with other units, it contains a dedicated displacement adder rather than sharing the integer calculation adder. The address unit contains a 32-entry data translation look-aside buffer. Entries can be used to translate single pages or groups of contiguous pages. The unit allows ranges of 8 Kbytes, 64 Kbytes, 512 Kbytes, or 4 Mbytes for each entry. The address unit can process up to three outstanding load misses to avoid blocking nondependent instructions.

Store instructions aggregate data in a 4-entry × 32-byte write buffer. The write buffer reduces off-chip bandwidth requirements by merging data from adjacent stores. It also allows early service for critical load data by temporarily delaying stores that would have otherwise occupied the data bus. The AXP architecture allows this reordering to improve performance; the memory barrier instruction can inhibit the reordering when necessary. The address unit allows back-to-back load-and-store operations in any order by accessing the current store tag with the last store data in separate cache tag and data arrays. The address unit supports wrapped reads (target word first) on primary cache misses, while filling 32-byte cache blocks. This minimizes the latency incurred when the return data is immediately needed. If the pipeline was blocked waiting for load data, it can continue as soon as the target word is returned at the same time that it fills the remainder of the cache line in the background.

Pipeline control/exceptions. The pipeline can be interrupted for a number of reasons including branch mispredictions, instruction cache misses, and interruption and exception conditions. Either a conditional branch or calculated jump instruction can produce branch mispredictions. They do not require hardware unrolling because both mispredictions are detected before the write back stage. Interruptions and exceptions cause traps to PALcode. These traps resemble mispredictions but also drain the pipeline before executing the new flow. Hardware reduces the idle pipeline time by overlapping the drain with the prefetch of the new instructions.

Privileged Architecture Library. A unique feature of the AXP architecture is the privileged architecture library. The PAL routines used with the 21064 allow flexibility in the definition of a hardware/software interface by assisting some hardware-related tasks and completely emulating others. For example, the hardware traps to PALcode to parse and service interruptions as well as to update translation look-aside buffers.

A second method of entering PALcode is through explicit CALL_PAL instructions. The chip supports 128 direct hardware dispatches for individual CALL_PAL-type instructions. Upon execution, a CALL_PAL instruction both branches to the selected PAL routine and enables PALmode privileges. These privileges allow PAL routines to access a complete internal state, otherwise hidden from the architected hardware/software interface. PAL can physically access both instruction and data-stream memory by disabling memory mapping, and assure atomic sequences of instructions by disabling interruptions. CALL_PAL routines support a variety of operations, generally too complex to be implemented in hardware. For example, the PAL provides the swap process context operation as a CALL_PAL instruction that can be unique for each operating system.

```
SUBQ    R0, R5, R1      ; cycle 0
ADDQ    R1, R2, R3      ; cycle 1
CMPLT   R3, #500, R4    ; cycle 2
BEQ     R4, Target      ; cycle 3
```

Figure 6. Multiple bypass paths allow many instructions to execute in sequential cycles despite the deep pipeline.

PALcode routines can be completely customized because they use a superset of the AXP instruction set. The architecture exclusively reserves five operation codes for PAL which allows each implementation to define these instructions for best use. A hardware implementation and PAL routines form a matched set that together make up the operating system and programmer interface. Since only the interface must remain consistent between implementations, future chips have complete flexibility to make low-level hardware trade-offs without impacting existing code. In addition, designers can redefine this interface to meet the needs of each operating system. The 21064 currently supports three operating systems with individual interface requirements.

Performance tuning. In the first production implementation of any new architecture, performance feedback is important for both software tuning and future hardware projects. With improving integration and cycle times, this information is increasingly difficult to obtain at the pin interface, or is so far removed from program execution that it is of little value. The 21064 contains two methods of providing more relevant information directly from running systems.

First, the Alpha AXP architecture offers a cycle counter capable of recording absolute and process virtual times at very fine intervals (single cycle on 21064). Its use, however, requires code modification. Second, the 21064 contains on-chip performance counters that count selected events and produce interruptions upon counter overflow. Through the use of PALcode and operating system utilities, these counters can collect data on unmodified applications. The design provides two counters that can select from a variety of sources

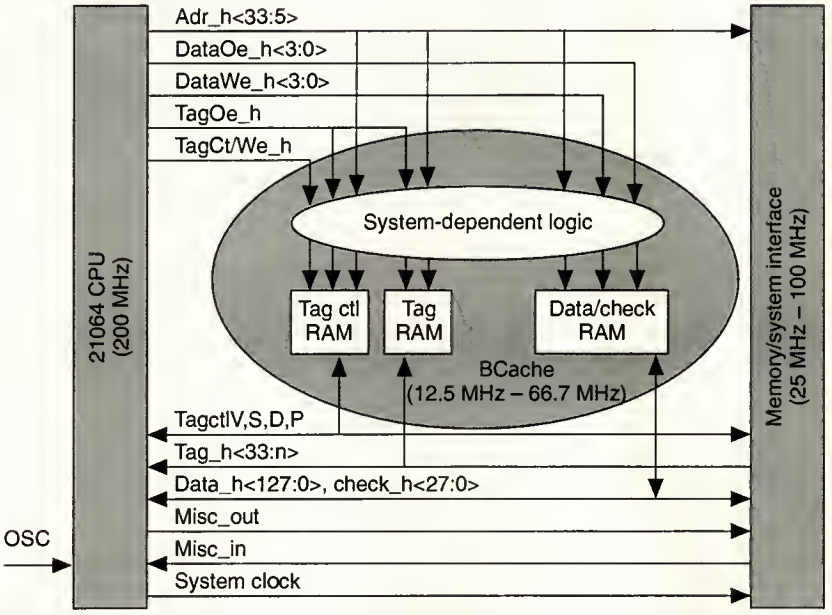


Figure 7. Chip interface. Three time domains exist in a typical system, with the CPU executing out of the internal caches at up to 200 MHz, the backup cache loop ranging from one third to one sixteenth of the CPU frequency, and the remaining system logic executing at one half to one eighth of the CPU frequency.

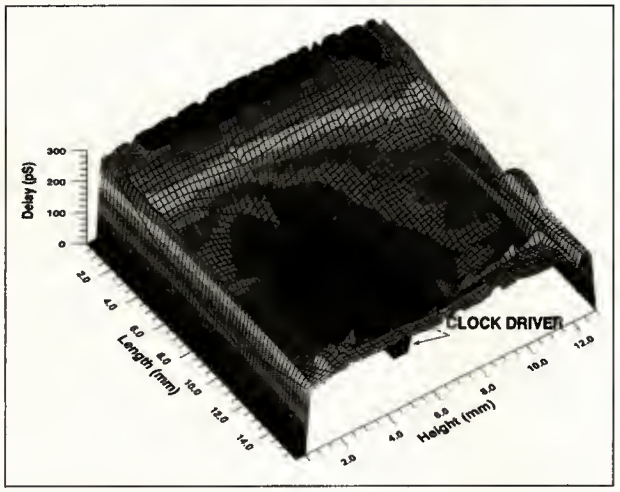


Figure 8. Alpha clock delay.

including instruction issue, pipeline stalls, and instruction mix as well as cache miss, branch misprediction, and two input pins that can be further broken down to gather external system data.

Interface. To accommodate a range of system designs, the interface is extremely flexible. See Figure 7 for a drawing of the chip interface. Although the chip operates with a 3.3-volt power supply, it can also interface with more common 5-volt logic. Data bus widths of 128 and 64 bits for reduced-cost systems are available, and the system clock speed can be set at any submultiple of the CPU speed from one half to one eighth. We have designed a 25-MHz EISA bus-based system that uses a one-to-six CPU clock divisor using standard PC interface parts. Even at 150-MHz CPU operation, cooling only requires a heat sink.

The chip supports up to 16 Gbytes of physical memory and an optional second-level back-up cache ranging in size from 128 Kbytes up to 16 Mbytes. The cache access path is combinatorial. It can support a variety of static RAM speeds, selectable through an on-chip register at 3 to 16 times the CPU clock cycle time. The interface supports parity or ECC protection. In the event of a back-up cache miss, the chip issues a system command to perform the necessary read or write operation. System commands interact with board logic in a handshake manner and operate at the selected system clock multiple, not the CPU clock speed.

Multiprocessing. The chip provides multiprocessing support in a flexible manner. Through valid, dirty, and shared (write protected) tag control signals, we can configure a write-back external cache on the 21064 to support a variety of cache coherence policies. Digital's systems use a conditional write-through policy although the chip can also support an ownership policy. Internal cache invalidate controls allow a system to maintain coherence of the internal cache. Through pin support for maintaining a backmap, the system can also implement cache invalidate filtering based on the contents of the primary cache if desired.

Clocking. Since it operates at speeds of up to 200 MHz, designing the 21064 required us to rethink many aspects of CMOS circuit design. Most critical was the decision to use a single-wire, two-phase clocking scheme. This type of clocking helps to eliminate dead time between phases. To ensure correct latching operation, though, the clock edge rate had to be extremely fast to avoid race-through of the latch data. Our solution included a very large clock driver with a final stage containing 10-11/64-inch-wide PMOS and 4-5/64-inch-wide NMOS devices. The driver switches the clock load in

Table 2. Measured performance under OpenVMS AXP V1.

	DEC 3000 Model 400	DEC 3000 Model 500	DEC 4000 Model 610	DEC 7000 Model 610	DEC 10000 Model 610
CPU frequency (MHz)	133	150	160	182	200
BCache size (Kbytes)	512	512	1,000	4,000	4,000
TPC-A (Rdb v.6)*	NA	NA	NA	302	NA
SPECint92	63.8	72.6	81.2	94.8	104.3
SPECfp92	112.2	126.0	143.1	182.1	200.4
SPECrate_int92	NA	NA	NA	NA	NA
SPECrate_fp92	2,631.6	2,967.4	3,317.1	4,126.0	
2 process			6,214.5	8,135.1	
3 process				11,859.8	
4 process				15,739.4	17,187.2
Linpack 100 × 100**	26.4	30.2	36.3	38.6	42.5
1,000 × 1,000**	90	107	114	141	155
Perfect BM suite†	18.1	20.4	22.9	26.0	28.6
Cernlib (CERN units)	16.9	19.0	21.0	23.6	26.0
Livermore loops	18.7	21.3	22.9	25.6	28.1
Slolom patches	5,644	6,022	6,384	7,018	7,248
SPECint89	65.8	73.5	83.7	95.1	104.5
SPECfp89	150.6	169.9	188.4	244.2	268.6
SPECmark89	108.1	121.5	136.2	167.4	184.1

* Transactions per second
** Project linear scaling for microprocessor configurations
† Geometric mean

0.5 ns, drawing a peak switching current of 43A. We extensively analyzed the clock both to ensure the integrity of the supply voltage during switching and to guarantee that the adjacent latches saw very little clock skew for proper operation. To address the supply voltage problem, we added 0.13 µF of on-chip decoupling capacitance. This was sufficient to supply all the charge associated with a complete CPU cycle with only 10-percent degradation of the supply voltage.

The skew problem required analysis of the 1.2-million-element RC clock grid. We used a simulator derived from the Carnegie-Mellon AWEsim circuit simulation program to examine the grid at 10-ps intervals. As shown in Figure 8, a monotonic clock wave propagates outward from the center clock driver. Any inward movement of the wave or large discrepancies would indicate potential timing hazards in the design. Such analysis proved necessary for correct operation of the chip, as early simulation results did, in fact, identify errors in the grid connection.

System performance

Performance tuning is an ongoing effort with work continuing in both compiler algorithms and optimizations as well as system tuning. However, as shown in Tables 2 and 3 (next page), initial data demonstrate excellent performance. These tables include results over a variety of commonly used benchmarks under both OpenVMS AXP V1 and DEC OSF/1 V1.2. Both SPECint92 and SPECfp92 values establish a new high point for system performance. Linpack among other floating-point intensive benchmarks demonstrates impressive floating-point capability, with the 1,000 × 1,000 values representing greater than 75 percent of the peak theoretical rate. Integer performance is equally impressive—the DEC 3000 Model 500X workstation achieves a SPECint92 rating of over 110. The more recent SPECrate benchmarks show nearly linear scaling across all multiprocessor configurations as demonstrated by the SPECrate data run under OpenVMS. *[For a fuller description of SPEC benchmarks, see H.G. Sachs et al., "Design and Implementation Trade-offs in the Clipper 400 Architecture," IEEE Micro, Vol. 11, No. 3, June 1991, pp. 18-21, 74-80.—Ed.]*

Table 4 shows results of benchmark performance for translated VAX images. The goal of the translation effort was to match or exceed the performance of similarly priced VAX systems. We met this goal, and many VAX user applications

Table 3. Measured performance under DEC OSF/1 V1.2.

	DEC 3000 Model 400	DEC 3000 Model 500	DEC 3000 Model 500X	DEC 4000 Model 610	DEC 7000 Model 610	DEC 10000 Model 610
CPU frequency (MHz)	133	150	200	160	182	200
BCache size (Kbytes)	512	512	512	1,000	4,000	4,000
SPECint92	74.7	84.4	110.9	94.6	103.1	116.5
SPECfp92	112.5	127.7	164.1	137.6	176.0	193.6
SPECrate-int92	1,763.0	1,997.0	2,611.0	2,198.0	2,571.7	2,765.0
SPECrate-fp92	2,662.0	3,023.0	3,910.0	3,247.0	4,178.7	4,368.4
Linpack 100 × 100	26.0	29.6	39.8	35.0	36.9	40.5
1,000 × 1,000*	91.7	103.5	133.2	110.1	137.8	151.1
X11perf (2D Kvec/s)	579.0	662	670	NA	NA	NA
X11perf (2D Mpix/s)	27.2	31.0	31.0	NA	NA	NA
Dhrystones/s V1.1	235,939	266,487	349,785	297,345	330,577	363,743
V2.1	238,095	263,157	333,333	294,117	333,333	357,142
Perfect BM suite**	18.4	20.7	26.2	23.1	26.4	29.2
Cernlib (CERN units)	18.8	21.3	28.9	23.2	26.0	29.0
Livermore loops**	17.4	19.5	26.3	22.3	25.4	27.8
Slalom patches	5,776	6,084	7,134	6,496	6,902	7,248
SPECint89	73.1	83.1	108.6	92.9	107.4	116.2
SPECfp89	141.7	162.8	208.9	177.0	249.8	275.8
SPECmark89	111.1	126.1	160.8	137.3	175.5	192.1

*64 bit, double precision
**Geometric mean

Table 4. Measured performance of VAX translated code under OpenVMS.

	DEC 7000 Model 610 Alpha AXP*	VAX 7000/610	DEC 3000 Model 500 Alpha AXP*	VAX 4000/90
SPECmark-89	44.43	42.09	34.37	32.77
SPECint-89	26.71	31.48	20.74	26.71
SPECfp-89	62.36	51.08	48.14	37.55

*Translated with DECmigrate

report between 10 to 15 percent faster times running the translated image on the AXP platform.

Table 5 shows results for translated MIPS images using DECmigrate. As can be seen, performance of the translated image approaches that of native compiled code on the DEC 3000 Model 500 system running DEC OSF/1.

THE ALPHA AXP ARCHITECTURE marks a new beginning for Digital. The combination of binary translation and PALcode affords the luxury of starting fresh, while maintaining strong compatibility with existing code. The architecture provides for growth in multiple fields as well as flexibility in the implementation of exception handling and operating system specific state. We carefully considered trends in computing such as multiple-instruction issue and multiprocessing to avoid restrictive requirements on future systems. Finally, 64-bit data capability and a 64-bit linear address space, offering over four billion times the range of a 32-bit space, should provide ample power and programming flexibility for years to come.

The goal of the chip design was to deliver the highest performance single-chip microprocessor in the industry, capable of forming the core of a range of systems from PC class to high-end server. Benchmark results attest to that accomplish-

ment. A wide range of system designs are currently available and expanding. In fact, the 21064 chip also forms the basis for the announced massively-parallel processing (MPP) supercomputer from Cray Research, Inc. With the availability of Microsoft Windows NT and native Novell NetWare, the architecture will offer an easy bridge for adding PC applications to the growing list of over 2,500 OpenVMS and Unix applications available today.

Unlike our previous architectures, Alpha AXP is an open computer architecture. Mitsubishi Electric Corp. recently joined over 35 other corporate Alpha AXP partners at all levels of design integration and will offer a second source for the 21064 as well as new designs in the future. Within Digital, we are currently designing multiple chips. High-performance parts include a speed enhanced version of the 21064 that will double the internal cache sizes and a next-generation, quad-issue processor. Design of a high integration device for reduced system cost is also in progress.

With the demonstrated performance of the current designs, availability of software, and a growing list of suppliers, the Alpha AXP architecture is well positioned for the future.

Table 5. MIPS translated code performance on DEC 3000 Model 500 (DEC OSF/1).

	Native	DECmigrate
SPECint92	84.4	68.6 (81%)
SPECfp92	127.7	81.9 (64%)

Acknowledgments

Many people contributed to the design of the Alpha AXP architecture, with a large majority of the current definition attributable to principal architects Dick Sites and Rich Witek. The 21064 design and verification team consisted of Dan Dobberpuhl, Rich Witek, Randy Allmon, Rob Anglin, Dave Bertucci, Sharon Britton, Linda Chao, Rob Conrad, Dan Dever, Bruce Gieseke, Soha Hassoun, Greg Hoeppner, John Kowaleski, Kathy Kuchler, Maureen Ladd, Mike Leary, Liam Madden, Ed McLellan, Dirk Meyer, Jim Montanaro, Don Priore, Vidya Rajagopalan, Sri Samudrala, Sri Santhanam, Scott Kreider, Stephan Meier, Andy Payne, Homayoon Akhiani, Mike Kantrowitz, and Dave Conroy, as well as others, who devoted tremendous amounts of their time and talents to ensure the successful completion of the design. Jim Montanaro, Mark Coiley, John Faricelli, and John Kowaleski created the clock analysis and graphical output.

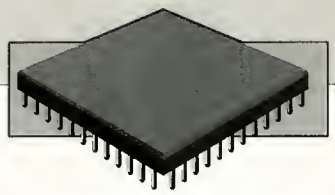
References

1. R.L. Sites, "Alpha AXP Architecture," *Comm. ACM*, Vol. 36, No. 2, Feb. 1993, pp. 33-44.
2. *Alpha Architecture Reference Manual*, R.L. Sites, ed., Digital Press, Bedford, Mass., 1992.
3. R. Conrad et al., "A 50-MIPS (Peak) 32/64-Bit Microprocessor," *Digest Tech. Papers, IEEE Solid-State Circuits Conf.*, Piscataway, N.J., Feb. 1989, pp. 76-77.
4. *VAX Architecture Reference Manual, Second Ed.*, R. Brunner, ed., Digital Press, Bedford, Mass., 1991.
5. *Cray-1 Computer System Reference Manual*, Form 2240004, Cray Research, Inc., 1977.
6. D. Dobberpuhl et al., "A 200-MHz 64-Bit Dual-Issue CMOS Microprocessor," *IEEE J. Solid-State Circuits*, Vol. 27, No. 11, Nov. 1992, pp. 1555-1567.
7. D. Dobberpuhl et al., "A 200-MHz 64-Bit Dual-Issue CMOS Microprocessor," *Digest Tech. Papers, IEEE Solid-State Circuits Conf.*, Feb. 1992, pp. 106-107.
8. *DECchip 21064-AA Microprocessor Hardware Reference Manual*, Digital Press, Bedford, Mass., October 1992.
9. R.L. Sites et al., "Binary Translation," *Comm. ACM*, Vol. 36, No. 2, Feb. 1993, pp. 69-81.

Edward McLellan is a principal engineer in Digital's Semiconductor Engineering Group. He has contributed to multiple chip designs, including PDP-11 and VAX floating-point processors, prior to the Alpha project. He was a coarchitect of the 21064 processor.

McLellan received a BS in computer and systems engineering from Renssalaer Polytechnic Institute and has done graduate work at the University of Massachusetts. Currently, he is working on a third-generation Alpha AXP processor.

Direct any questions concerning this article to the author at the Digital Equipment Corporation, HLO2-3/J03, Hudson, MA 01749; mclellan@ad.enet.dec.com.



Sparcle: An Evolutionary Processor Design for Large-Scale Multiprocessors

Working jointly at MIT, LSI Logic, and Sun Microsystems, designers created the Sparcle processing chip by evolving an existing RISC architecture toward a processor suited for large-scale multiprocessors. This chip supports three multiprocessor mechanisms: fast context switching, fast, user-level message handling, and fine-grain synchronization. The Sparcle effort demonstrates that RISC architectures coupled with a communications and memory management unit do not require major architectural changes to support multiprocessing efficiently.

Anant Agarwal

John Kubiatowicz

David Kranz

Beng-Hong Lim

Donald Yeung

Massachusetts Institute of Technology

Godfrey D'Souza

LSI Logic

Mike Parkin

Sun Microsystems

The Sparcle chip clocks at no more than 40 MHz, has no more than 200,000 transistors, does not use the latest technologies, and dissipates a paltry 2 watts. It has no on-chip cache, no fancy pads, and only 207 pins. It does not even support multiple-instruction issue. Then why do we think this chip is interesting?

Sparcle is a processor chip designed to support large-scale multiprocessing. We designed its mechanisms and interfaces to provide fast message handling, latency tolerance, and fine-grain synchronization. Specifically, Sparcle implements

- *Mechanisms to tolerate memory and communication latencies, as well as synchronization latencies.* Long latencies are inevitable in large-scale multiprocessors, but current microprocessor designs are ill-suited to handle such latencies.
- *Mechanisms to support fine-grain synchronization.* Modern microprocessors pay scant attention to this aspect of multiprocessing, usually providing just a test-and-set instruction, and in some cases, not even that.
- *Mechanisms to initiate communication actions to remote processors across the communications network, and to respond rapidly to asynchronous events such as synchronization faults and message arrivals.* Current microprocessor designs do not support a clean communications interface between the processor and the communications network. Furthermore, traps and other asynchronous event-handlers are inefficient on many current microprocessors, often requiring tens of cycles to reach the appropriate trap service routine.

The impetus for the Sparcle chip project was our belief that we could implement a processor that provides interfaces for the above mechanisms by making small modifications to an existing microprocessor. Indeed, we derived Sparcle from Sparc[1] (scalable programmable architecture from Sun Microsystems), and we integrated it into Alewife,[2,3] a large-scale multiprocessor system being developed at MIT.

Sparcle tolerates long communication and synchronization latencies by rapidly switching to other threads of computation. The current implementation of Sparcle can switch to another thread of computation in 14 cycles. Slightly more aggressive modifications could reduce this number to four cycles. Sparcle switches to another thread when a cache miss that requires service over the communications network occurs, or when a synchronization fault occurs. Such a processor requires a pipelined memory and communications

0272-1732/93/0600-0048$03.00 © 1993 IEEE

system. In our system, a separate communications and memory management chip (CMMU) interfaces to Sparcle to provide the desired pipelined system interface. Our system also provides a software prefetch instruction. For a description of the modifications to a modern RISC microprocessor needed to achieve fast context switching, see our discussion under architecture and implementation of Sparcle later in the article.

Sparcle supports fine-grain data-level synchronization through the use of full/empty bits, as in the HEP computer.[4] With full/empty bits, a lock and access of the data word protected by the lock can be probed in one operation. If the synchronization attempt fails, the synchronization trap invokes a fault handler. In our system, the external communications chip detects synchronization faults and alerts Sparcle by raising a trap line. The system then handles the fault in software trap code.

Finally, Sparcle supports a highly streamlined network interface with the ability to launch and receive interconnection network messages. While this design implements the communications interface with the interconnection network in a separate chip, the CMMU, future implementations can integrate this functionality into the processor chip. Sparcle supports rapid response to asynchronous events by streamlining Sparc's trap interface and by supporting rapid dispatch to the appropriate trap handler. To achieve this, Sparcle provides two special trap lines for the most common types of events—cache misses to remote nodes and synchronization faults. Sparcle uses a third trap line for all other types of events. Also, this chip has an increased number of instructions in each trap dispatch entry so that vital trap codes can be put in line at the dispatch points.

Sparcle's design process was unusual in that it did not involve developing a completely new architecture. Rather, we implemented Sparcle with the help of LSI Logic and Sun Microsystems by slightly modifying the existing Sparc architecture. At MIT, we received working Sparcle chips from LSI Logic on March 11, 1992. These chips have already undergone complete functional testing. We are currently continuing to implement the Alewife multiprocessor so that we can thoroughly evaluate our ideas and subject the Sparcle chips to full-speed testing. Figure 1 shows an Alewife node with the Sparcle chip.

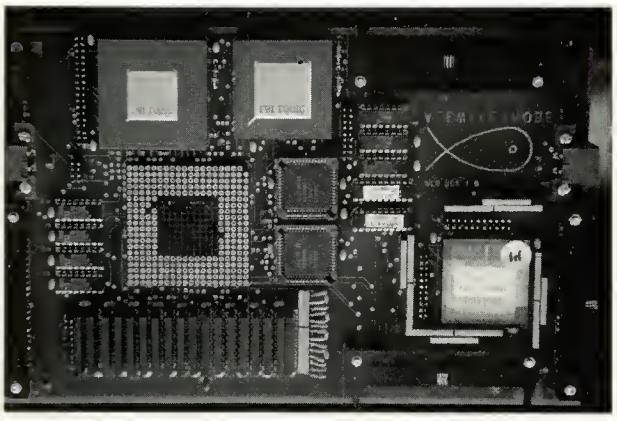


Figure 1. An Alewife node.

Mechanisms for multiprocessors

By supporting the widely used shared-memory and message-passing programming models, Sparcle eases the programmer's job and enhances parallel program performance. We have implemented programming constructs in parallel versions of Lisp and C that use these features. Sparcle's features fall into three areas, the first two of which support the shared-memory model:

- **Fine-grain computation.** Efficient support of fine-grain expression of parallelism and synchronization can enhance performance by increasing parallelism and reducing communication overhead. This enhancement relieves the programmer of undue effort in partitioning data and controlling flow into coarser chunks to increase performance.
- **Memory latency tolerance.** Context switching and data prefetching can reduce communication overhead introduced by network delays. For shared-memory programs, the switch must be very fast and occur automatically when a remote cache miss occurs.
- **Efficient message interface**. The ability to send and receive messages is needed to support message-passing programs. Such interfacing can also improve the performance of shared-memory programs in some common situations.

Before we can examine the implementation of these features in Sparcle, we need to consider each of these areas in turn, and discuss why they are useful for large-scale multiprocessing.

Fine-grain computation. As multiprocessors become larger, the grain size of parallel computations decreases to satisfy higher parallelism requirements. Computational grain size refers to the amount of computation between synchronization operations. Given a fixed problem size, the overhead of parallel and synchronization operations limits the ability to use a larger number of processors to speed up a program. Systems supporting fine-grain parallelism and synchronization attempt to minimize this overhead so that parallel programs can achieve better performance.

The challenge of supporting fine-grain computation is in implementing efficient parallelism and synchronization constructs without incurring extensive hardware cost, and without reducing coarse-grain performance. By taking an evolutionary approach in designing Sparcle, we have attempted to satisfy these requirements.

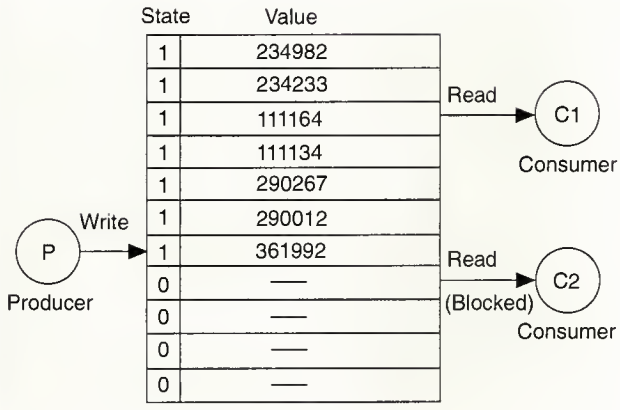


Figure 2. J-structures.

We can express fine-grain parallelism and synchronization at the data level (data-level parallelism) or at the thread level (control-level parallelism).

Data-level parallelism. Data-level parallelism and synchronization allows the program to synchronize at the level of the smallest possible unit—a memory word. At the programming language level, we provide parallel do-loops to express data-level parallelism, and J-structure and L-structure arrays to express fine-grain data-level synchronization.

Inspired by the I-structures of Arvind, Nikhil, and Pingali,[5] the J-structure is a data structure for producer-consumer style synchronization. It is like an array, but each element has an additional state—full or empty. The initial state of a J-structure element is empty. A reader of an element waits until the element's state is full before returning the value. A writer of an element writes a value, sets the state to full, and signals waiting readers to proceed. A write to a full element signals an error. For efficient memory allocation and cache performance, J-structure elements can be reset to an empty state. Figure 2 illustrates how J-structures can be used for data-level synchronization.

In the example of Figure 2, producer P is sequentially filling in the elements of a J-structure. Consumer C1 reads an element that is already filled and immediately gets its value. Consumer C2 reads an empty element and thus has to wait for P to write the element. Since we are synchronizing at the level of individual elements, both C1 and C2 can access the elements of the J-structure without waiting for P to completely fill all the elements of the J-structure.

L-structures are similar to J-structures but support three operations: a locking read, a nonlocking read, and a synchronizing write. A locking read waits until the element is full before emptying it (that is, locking it) and returning the value. A nonlocking read operation also waits until the element is full, but returns the value without emptying the element. A synchronizing write stores a value to an empty element and sets it to full, releasing any waiters. An L-structure thus allows mutually exclusive access to each of its elements and allows multiple nonlocking readers.

Sparcle supports J- and L-structures, as well as other types of fine-grain data-level synchronization, with per-word, full/empty bits in memory.[4] Sparcle provides new load/store instructions that interact with the full/empty bits. The design also includes an extra synchronous trap line to deliver the full/empty trap. This extra line allows Sparcle to immediately identify the trap.

Control-level parallelism. Control-level parallelism may be expressed by wrapping *future* around an expression or statement *X*. The *future* keyword declares that *X* and the continuation of the future expression may be evaluated concurrently. Fine-grain support allows the amount of computation needed for evaluating *X* to be small without severely affecting performance.

If the compiler or runtime system chooses to create a new task to evaluate *X*, it also creates an object known as a *placeholder* that is returned as the value of the future expression. The placeholder is created in an undetermined state. Evaluation of *X* yields its value and determines the placeholder. Any task that attempts to use the value of *X* before *X* has been completely evaluated will encounter the undetermined placeholder and will suspend operation until the placeholder is determined.

This functionality is implemented using (by software convention) the low bit of a data value as a placeholder tag; that is, a pointer to a placeholder has the low bit set and all other values have the low bit clear. New add, subtract, and compare instructions in Sparcle trap if the low bit of any operand is set. Likewise, dereferencing a pointer with the low bit set will cause an address alignment trap to a similar routine. If the trap handler can determine the value at the placeholders, it places this value in the target register, and normal execution resumes. Otherwise, the trapping task waits until the value of the placeholder becomes available.

With this support, a compiler can generate code without knowing which data values may be computed concurrently. Consequently, Sparcle incurs no runtime overhead to ensure the detection of placeholders.

Memory latency tolerance. Since memory in large-scale multiprocessors is distributed, cache misses to remote locations will incur long latencies and potentially reduce processor use. Figure 3 illustrates this problem by depicting processor and network activity when a single thread executes on the processor. When the thread suffers a long-latency cache miss, the processor waits for the miss to be satisfied before it can proceed. While waiting, both the processor and the network suffer idle time, thereby reducing their effective usage. Using latency tolerance mechanisms alleviates this problem and helps improve processor and network usage.

The general class of latency tolerance solutions all implement mechanisms that allow multiple outstanding memory transactions and can be viewed as a way of pipelining the processor and the network. The key difference between this pipeline into the network and the processor's execution pipeline is that the latency associated with the communication pipeline cannot be predicted easily at compile time. A compiler then has difficulty scheduling operations for maximal resource use. Systems must implement dynamic pipelines into the network in which the hardware ensures that multiple, previously issued memory operations have completed before issuing operations that depend on their completion. Context switching is one mechanism for dynamic pipelining. Other methods include prefetching and weak ordering.[6-8]

Sparcle implements fast context switching as its primary mechanism for dynamic latency tolerance. (Sparcle and its memory controller provide nonbinding prefetch instructions as well.) As illustrated in Figure 4, the basic idea is to overlap the latency of a memory request from a given thread of computation with the execution of a different thread. In the figure, when thread 1 suffers a cache miss, the processor switches to thread 2, thereby overlapping the cache miss latency of thread 1 with useful computation from thread 2.

In Alewife, when a thread issues a remote transaction or suffers an unsuccessful synchronization attempt, the Alewife CMMU traps the processor. If the trap resulted from a cache miss to a remote node, the trap handler forces a context switch to a different thread. Otherwise, if the trap resulted from a synchronization fault, the trap handling routine can switch to a different thread of computation. For synchronization faults, the trap handler might also choose to retry the request immediately (spin).

Processors that switch rapidly between multiple threads of computation are called multithreaded architectures. The prototypical multithreaded machine is the HEP. In the HEP, the processor switches every cycle between eight processor-resident threads. Cycle-by-cycle interleaving of threads is termed fine multithreading. Although fine multithreading offers the potential for high processor usage, it results in relatively poor single-thread performance and low processor use when there is not enough parallelism to fill all the hardware contexts.

In contrast, Sparcle employs block multithreading or coarse multithreading. That is, context switches occur only when a thread executes a memory request that must be serviced by a remote node in the multiprocessor, or on a failed synchronization request. Thus, a given thread continues to execute as long as its memory requests hit in the cache or can be serviced by a local memory module, and as long as synchronization attempts are successful. Block multithreading thus allows a single thread to benefit from the maximum performance of the processor. For multithreading to be useful in tolerating latency, however, the time required to switch to another thread must be shorter than the time to service a remote request. This requires multiple register sets or some other hardware-supported mechanism.

Efficient message interface. An efficient message interface that allows the processor to access the interconnection network directly makes some parallel operations significantly more efficient than if they were implemented solely with shared-memory operations. Examples include remote thread creation and barrier synchronization. With a fast message in Alewife, we can create a thread on a remote processor in 7 μs. Restricting ourselves to shared-memory operations, remote thread creation takes 24 μs. Kranz and associates[9] have studied the importance of an efficient message interface in a shared-memory setting.

In Sparcle, we accomplish a fast message send operation by using the cache bus and coprocessor interface to store data in registers directly into the network, and to load data from the network directly into registers. Two new load/store instructions handle the loading and storing. Sparcle also supports direct memory access for larger messages.

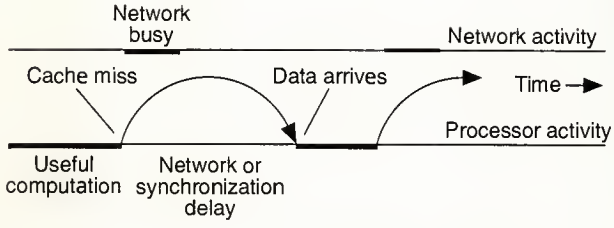


Figure 3. Processor and network activity when a single thread executes on the processor and no latency tolerance mechanisms are employed.

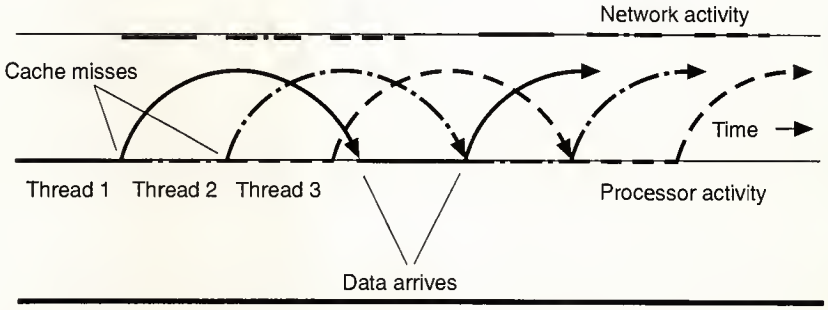


Figure 4. Processor and network activity when multiple threads execute on the processor and fast context switching is used for latency tolerance.

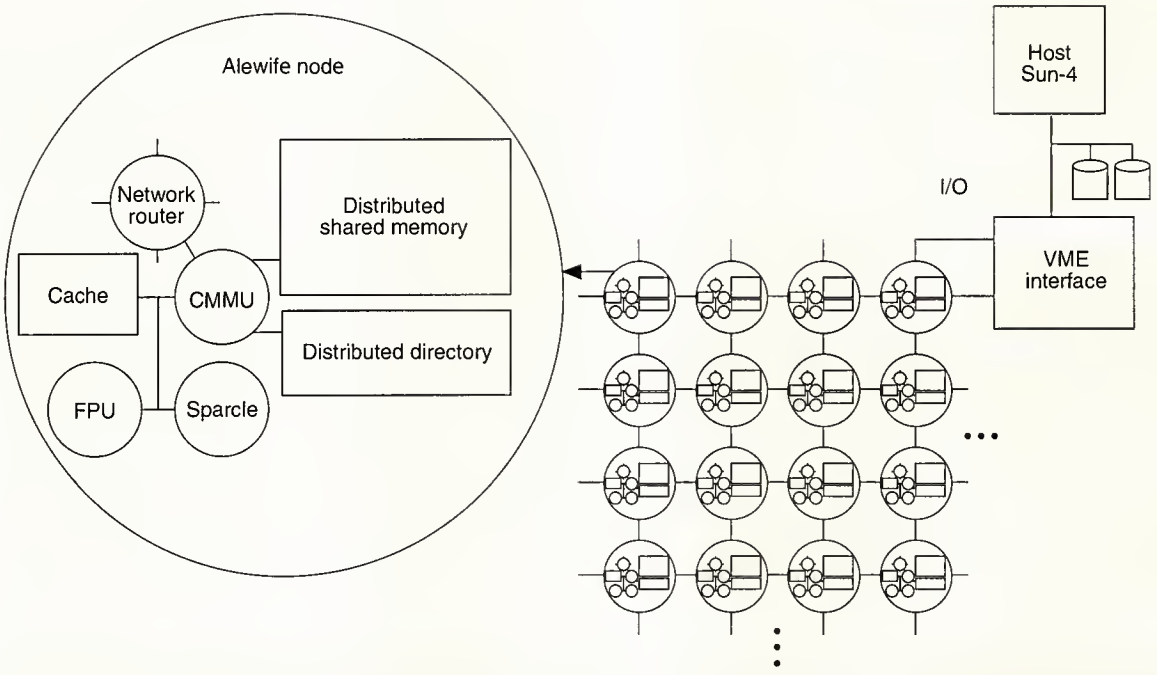


Figure 5. Structure of the Alewife machine.

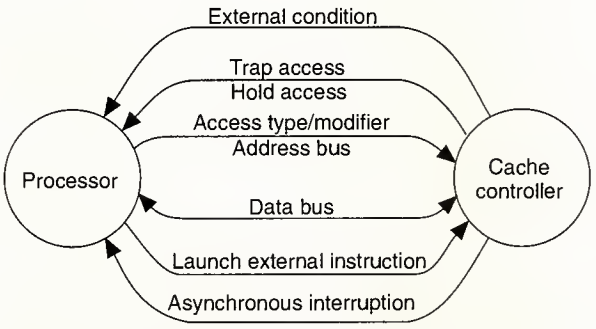


Figure 6. Interface between the processor pipeline and memory controller.

Alewife machine interfaces

The Sparcle chip is part of a complete multiprocessing system. It serves as the CPU for the Alewife machine[2]—a distributed shared-memory multiprocessor with up to 512 nodes and hardware-supported cache coherence. Figure 5 depicts the Alewife machine as a set of processing nodes connected in a mesh topology. Each Alewife node consists of a processor, a 64-Kbyte cache, a 4-Mbyte portion of globally-shared distributed memory, a CMMU, a floating-point coprocessor, and a network switch. An additional 4 Mbytes of local memory holds the coherence directory, code, and local data. The network switch chip is an Elko-series mesh routing chip (EMRC) from Caltech that has 8-bit channels. The network operates asynchronously with a switching delay of 30 ns per hop and 60 Mbytes/s through bidirectional channels.

The single-chip CMMU performs a number of tasks, including cache management, DRAM refresh and control, message queuing, remote memory access, and direct memory access. It also supports the LimitLESS cache-coherence protocol,[10] which maintains a few pointers per memory block in hardware (up to five in Alewife) and emulates additional pointers in software when needed. Through this protocol, all the caches in the system maintain a coherent view of global memory.

Sparcle implements a powerful and flexible interface to the CMMU. As depicted in Figure 6, this interface couples the processor pipeline with the CMMU. The interface can be divided into two general classes of signals: flexible data access mechanisms and flexible instruction extension mechanisms.

Together, the Access Type, Address Bus, Data Bus, and Hold Access line form the nucleus of data access mechanisms and comprise a standard external cache interface. To permit the construction of other types of data accesses for synchronization, we have supplemented this basic interface with three classes of signals:

- A *Modifier* that is part of the operation code for load/

store instructions and that is *not* interpreted by the core processor pipeline. The modifier provides several "flavors" of load/store instructions.
- Two *External Conditions* that return information about the last access. They can affect the flow of control through special branch instructions.
- Several vectored memory exception signals (denoted *Trap Access* in the figure). These synchronous trap lines can abort active load/store operations and can invoke function-specific trap handlers.

These mechanisms permit us to extend the load/store architecture of a simple RISC pipeline with a powerful set of operations.

An instruction extension mechanism permits us to augment the basic instruction set with external functional units. Instructions that are added in this way can be pipelined in the same fashion as standard instructions. To make this work, Sparcle reserves a special range of opcodes for external instructions. Also, the memory controller fetches new instructions from the cache bus at the same time that the processor does. Consequently, when the processor decodes an instruction in this range, it asserts the Launch External Inst signal, telling the CMMU to begin execution of the last fetched instruction. Note that the coprocessor interfaces of several microprocessors already provide this functionality.

We contend that we can design such a powerful interface between the processor pipeline and the communications and memory management hardware without significantly modifying the core RISC pipeline of contemporary processors. With this interface in mind, we first discuss several efficient multiprocessor mechanisms that are provided by the Sparcle processor. Later we touch upon the support which the memory controller must provide for these mechanisms.

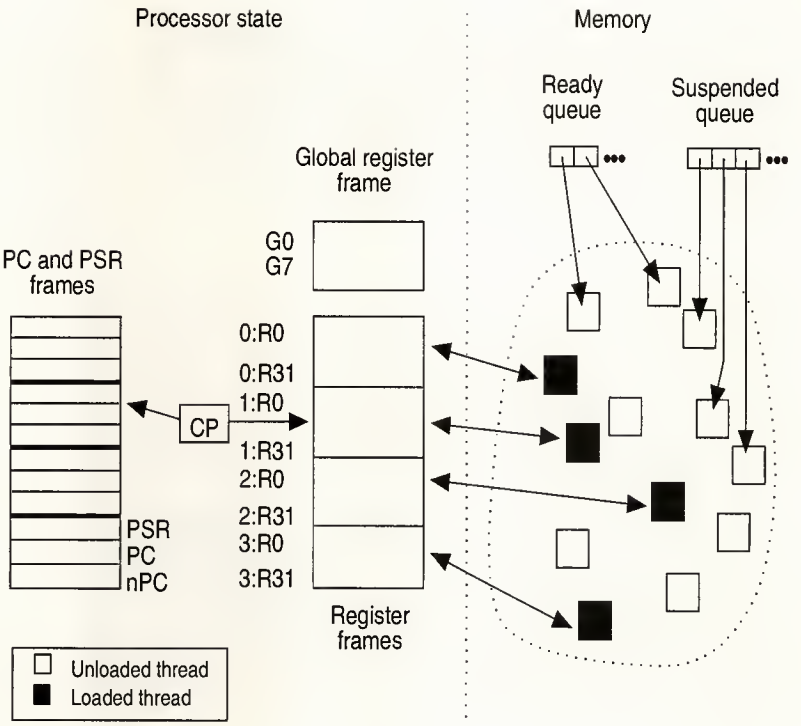


Figure 7. Block multithreading and virtual threads.

Sparcle architecture and implementation

Sparcle is best described as a conventional RISC microprocessor with a few additional features to support multiprocessing. These features include support for latency tolerance, support for fine-grain synchronization, and support for fast message handling. Before we describe how we implemented them in the Sparc processor, we need to discuss these features. Then we can indicate how they can also be implemented in other RISC microprocessors.

Mechanisms for latency tolerance. Figure 7 illustrates fast context switching on a generic processor. This diagram shows four separate register sets with associated program counters and status registers. Each register set represents a context. A hardware register called the context pointer or CP points to the active context. Consequently, a hardware context switch requires only that the context pointer be altered to point to another context. (Depending on details of the implementation, some number of cycles may be needed to flush the pipeline before executing a new context.) This figure also shows four threads actively loaded in the processor. These four threads are part of a much larger set of runnable and suspended threads that the runtime system maintains.

Implementation of fast context switching in Sparc. In a similar fashion, Sparcle uses multiple register sets to implement fast context switching. The particular Sparc design that we modified has eight overlapping register windows. Rather than using the register windows as a register stack, we used them in pairs to represent four independent, nonoverlapping contexts. We use one as a context for trap and message handlers, as described by Dally et al.[11] and Seitz et al.,[12] and the other three for user threads. The Sparc current window pointer (CWP) serves as the context pointer. Further, the window

```
RDPSR   R16          ; Save PSR in reserved register.
NEXTF   R0, R0, R0   ; Move to next active context.
WRPSR   R16          ; Restore PSR from other context.
JMPL    R17, R0      ; Restore PC
RETT    R18, R0      ; Restore nPC and return from trap.
```

Figure 8. Context switch trap code for Sparcle

Cycle	Operation
0	Fetch of data instruction (load or store)
1	Decode of data instruction (load or store)
2	Execute instruction (compute address)
3	Data cycle (which will fail)
→4	Pipeline freeze, indicate exception to processor
5	Pipeline flush (save PC)
6	Pipeline flush (save nPC, decrease CWP)
7	Fetch: RDPSR PSRREG (save PSR in reserved register)
8	Fetch: NEXTF (advance CWP to next active context using WIM)
9	Fetch: WRPSR PSRREG (restore PSR for new context)
10	Fetch: JMPL R17 (load PC, return from trap and)
11	Fetch: RETT R18 (reexecute trapping instruction)
12	Dead cycle from JMPL
13	First fetch of new instruction
14	Dead cycle from RETT (folded into switch time)

Figure 9. Anatomy of a context switch in Sparcle.

invalid mask (WIM) indicates which contexts are disabled and which are active. This particular use of register windows does not involve any modifications, just a change in software conventions.

Unfortunately, the Sparc processor does not have four sets of program counters and status registers. Since adding such facilities would impact the pipeline significantly, we implemented rapid context switching via a special trap with an extremely short trap handler. Thus, when the processor attempts to access a remote memory location that is not in the local cache, the CMMU causes a synchronous memory fault to Sparcle, while simultaneously sending a request for data to the remote node. The trap handler then saves the old program counter and status register, switches to a new context, restores a new program counter and status register, returns from the trap to begin execution in the new context.

With the goal of shortening this trap handler as much as possible, we made the following modifications to the Sparc architecture:

- So that the processor traps immediately to the context-switch code without having to decode the trap type, we added an extra synchronous trap line (with corresponding trap vector).
- We added a new instruction called NEXTF. It is much like the Sparc SAVE instruction except that the window pointer is advanced to the next active context as indicated by the window invalid mask register. If no additional contexts are active, it leaves the window pointer unchanged.
- We increased the number of instructions for each entry in the Sparc trap vector from 4 to 16. This allows the context switch and other small trap handlers to execute in the trap vector directly.
- We made the value of the current window pointer available on external pins. Among other things, this permits the emulation of multiple hardware contexts in the Sparc floating-point unit by modifying floating-point instructions in a context-dependent fashion as they are loaded into the FPU and by maintaining four different sets of condition bits. Consequently, the context-switch trap handler does not have to worry about the FPU.

Figure 8 shows the context-switch trap handler with these changes. When the trap occurs, Sparcle switches *one* window backward (as does a normal Sparc). This switch places the window pointer *between* active contexts, where the Alewife runtime system reserves a few registers for the context state. As with normal Sparc trapping behavior, the hardware writes the PC and nPC to registers R17 and R18. This trap code places the processor status register (PSR) in register R16.

As depicted in Figure 9, the net effect is that a Sparcle context switch takes 14 cycles. This illustrates the total penalty for a context-switch on a data instruction. Note that, while this diagram shows 15 cycles, one of them is the fetch of the first instruction from the next context.

By maintaining a separate PC and processor status register for each context, a more aggressive processor design could switch contexts much faster. However, even with 14 cycles of overhead and four processor-resident contexts, multithreading can significantly improve system performance.[13,14]

Support for fine-grain synchronization. As discussed earlier, fine-grain data-level synchronization is expressed with J- and L-structures and implemented using new instructions that interact with full/empty bits in memory. Sparc implements the new load, store, and swap instructions using the Sparc alternate address space instructions. We have modified these instructions in two ways:

1. The load, store, and swap alternate space instructions in Sparcle are unprivileged for ASI values in the range 0×80 to $0 \times FF$. They remain privileged for ASI values less than 0×80. The CMMU uses the ASI value as an extended opcode; that is, ASI 0×84 corresponds to the load and

trap if empty operation. This allows user code to interact directly with full/empty bits.

2. We have used several new opcodes to produce specific ASIs on the Sparcle output pins while allowing the register + offset addressing mode. The normal load/store ASI instructions only allow register + register addressing.

A new dedicated synchronous trap line carries full/empty trap signals. J- and L-structure operations are implemented with the following special load/store instructions:

LDN	Read location
LDEN	Read location and set to empty
LDT	Read location if full, else trap
LDET	Read location and set to empty if full, else trap
STN	Write location
STFN	Write location and set to full
STT	Write location if empty, else trap
STFT	Write location and set to full if empty, else trap

In addition to possible trapping behavior, each of these instructions sets a coprocessor condition code to the state of the full/empty bit at the time the instruction starts execution. Either trapping or an explicit test of this condition code will detect a synchronization failure. When a trap occurs, the trap handling software decides what action to take.

Implementation of J-structures. To demonstrate how the special load/store instructions can be used, we will describe how we implement J-structures and present the cycle counts for various synchronizing operations. Sparcle implements a J-structure allocation by allocating a block of memory with the full/empty bit for each word set to empty. Resetting a J-structure element involves setting the full/empty bit for that element to empty. Implementing a J-structure read operation is also straightforward: it is a memory read that traps if the full/empty bit is empty. Sparcle implements it with a single instruction:

```
LDT (R1),R2 ; R1 points to J-structure location
```

If the full/empty bit is empty, the reading thread may need to suspend execution and queue itself on a wait queue associated with the empty element. To minimize memory usage, we use a single memory location to represent *both* the value of the J-structure element and the wait queue. This implies that we need to associate two bits of state with each J-structure element: whether the element is full or empty and whether the wait queue is locked or not.

```
MOVE   $0, R3      ; set up swap register.
SWAPT  R3, (R1)    ; swap zero with J-structure location, trap if full.
CMP    $-1, R3     ; check if queue is empty.
BEG, a %done       ; branch if no waiters to wake up.
STFT   R2, (R1)    ; write value and set to full (delay slot).
  ⋮
<wake up waiters and store value>
  ⋮
%done
```

Figure 10. Machine code implementing a J-structure write.

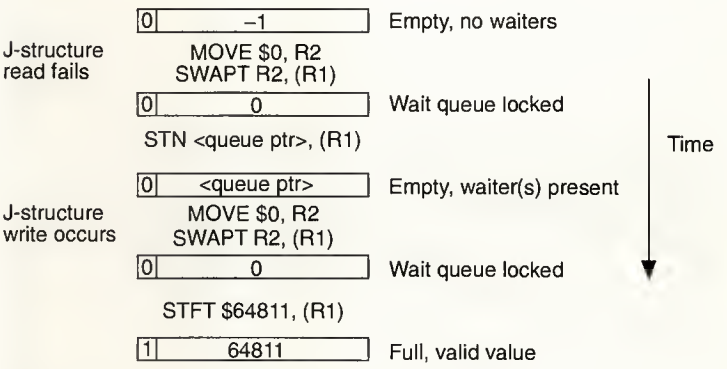


Figure 11. Reading and writing a J-structure slot.

Other architectures implement these two state bits directly in hardware by having multiple state bits per memory location.[15,16] Instead of providing an additional hardware bit, we take advantage of Sparc's atomic register-memory swap operation. Since the writer of a J-structure element knows that the element is empty before it does the write operation, it can use the atomic swap to synchronize access to the wait queue. With this approach, a single full/empty bit is sufficient for each J-structure element. A writer needs to check explicitly for waiters before undertaking the write operation.

Using atomic swap and full/empty bits, the machine code in Figure 10 implements a J-structure write. In this figure, R1 contains the address of the J-structure location to be written to, and R2 contains the value to be written. Also, –1 is the end of the queue marker, and 0 in an empty location means that the queue is locked. Compared with the hardware approach, this implementation costs an extra move, swap, compare, and branch to check for waiters. However, we believe that the reduction in hardware complexity is worth the extra instructions.

Figure 11 gives a scenario of accesses to a J-structure location under this implementation and illustrates the possible states of a J-structure slot. Here, R1 contains a pointer to the J-structure slot.

Table 1. Summary of fast-path costs of J-structure and L-structure operations, compared with normal array operations.

Element	Action	Instructions	Cycles
Array	Read	1	2
	Write	1	3
J-structure	Read	1	2
	Write	5	10
	Reset	1	3
L-structure	Read	2	5
	Write	5	10
	Peek	1	2

```
STIO    R2, $ipiout0 ; Store header.
STIO    R3, $ipiout1 ; Store data word.
STIO    R4, $ipiout2 ; Store address of data.
STIO    R5, $ipiout3 ; Store length of data.
IPILAUNCH 2, 1       ; Launch message. Descriptor is 2 double-
                     ; words long and contains 1 double-word
                     ; of explicit data (from R2 and R3).
```

Figure 12. Machine code implementing a message send.

Table 1 summarizes the instruction and cycle counts of J-structure and L-structure operations for the case where no waiting is needed on read operations and no waiters are present on write operations. In Sparcle, as in the LSI Logic Sparc, normal read operations take two cycles and normal write operations take three cycles, assuming cache hits. A locking read is considered a write and thus takes three cycles.

Support for futures and placeholders. To support futures and placeholders, Sparcle provides automatic and efficient detection and handling of placeholders via traps. Two Sparcle modifications are involved.

First, to detect placeholders, Sparcle adds two new instructions called NTADD and NTSUB. These instructions cause tag overflow traps whenever the low bit of either of their operands is set. (NTADD and NTSUB are modifications of the Sparc tagged instructions TADDCCTV and TSUBCCTV that trap whenever the low two bits of either of their operands are set.) As discussed earlier, only pointers to placeholders have the low bit set. With tag overflow traps, NTADD and NTSUB automatically detect placeholders in add, subtract, and compare operations. The address alignment trap in Sparcle detects placeholders in pointer dereferencing operations.

Second, to efficiently handle traps caused by placeholders, the trap vector number that is generated by tag overflow and address alignment traps depends on the register containing the placeholder. This feature saves the trap handler from having to waste cycles decoding the trapping instruction to find out which register contains the offending placeholder. Johnson[17] and Ungar et al.[18] have proposed similar mechanisms.

Fast message handling. Most distributed shared-memory machines are built on top of an underlying message-passing substrate. Traditional shared-memory machines provide a layer of hardware that implements some coherence protocol between the processor and the interconnection network. It is natural, then, to provide the processor with direct access to the network in addition to the shared-memory interface because many operations benefit greatly from direct network access. Sparcle supports sending and receiving messages via a memory-mapped interface to the interconnection network.

Send. Sparcle sends messages through a two-phase process: first describe, then launch. Sparcle composes a message by writing directly to the interconnection network queue using a special store instruction called STIO (for store IO). The queues are memory mapped as an array of network registers in the CMMU, called the output descriptor array. In terms of performance, write operations into this array incur the same cost as write hits into the cache.

The first word of the message must be a header indicating a message opcode and the destination node. Sparcle reserves a range of opcodes for privileged use by the operating system. The rest of the message can contain immediate values from registers, or address and length pairs which invoke DMA on blocks from memory.

After the message is composed, a coprocessor instruction launches the message. Figure 12 illustrates the sending of a single message with one data word and one block of data from memory. In addition to the required header, this message includes one explicit data word and one block of data from memory. On entry to this code sequence, register R2 contains the header, R3 contains the data word, R4 the address of the data block, and R5 the length of the data block. If Sparcle is in the user mode and the header is privileged, an exception will occur. The CMMU maintains the atomicity of messages as described in the next section.

Receive. A message arrival causes a trap. The trap handler can either load words directly from the incoming message into registers using a special load instruction called LDIO (for load IO) or initiate a DMA sequence to store the message into memory. If the latter option is chosen, the processor can direct the CMMU to generate an interrupt after the storeback is complete.

Support for message handling. The following features of Sparcle support messaging:

- Special user-level load/store instructions allow fast composition of outgoing messages and fast examination of

incoming messages. An ASI value is reserved for the transferring of data to and from message register values. This ASI is produced by two new Sparcle instructions, STIO and LDIO. Although these instructions support a memory-mapped interface to the network registers, addresses for the message queues fit completely into the address offset field. Consequently, the compiler can generate instructions that perform direct register-to-register moves between the processor and the network queues.

- Register windows permit fast processing of message interrupts. One of the four hardware contexts is reserved for message processing. Consequently, the message interrupt handler needs only to alter the current window pointer so that this special context is active. No registers need to be saved and restored.
- Coprocessor instructions for message launch and disposal permit pipelining of network operations. Further, opcode bits in the launch and disposal instructions contain information about the format of messages that are about to be sent or received into memory. Thus, message format is completely under control of the compiler. Finally, the coprocessor interface permits a precise identification of the commit point for launch instructions, ensuring that message launches are atomic.
- Fast interrupt operations allow rapid entry into message handler code on the arrival of a message. In our current implementation, because interrupts always force the processor into the supervisor mode, user-level receipt of messages requires a few extra cycles for the processor to transfer control to user code. In a more aggressive implementation, the processor would support a user-level return from trap.

The CMMU interface

From this discussion we can clearly see that the Sparcle processor is part of a complete system. Consequently, several of the mechanisms that were included in Sparcle are incomplete without the support of the CMMU. Here we briefly discuss the Alewife CMMU and how it interfaces to Sparcle. Although the Alewife CMMU provides a number of features, we focus on the cache controller and message interface.

Earlier, under Alewife machine interfaces, we discussed two categories of signals in the interface between processor and CMMU: flexible data access mechanisms and flexible instruction extension mechanisms. Figure 13 makes this interface more concrete by showing Sparcle equivalent names for all of the signals. Each signal in this figure corresponds directly to signals in Figure 6.

A few of the data access mechanisms require further discussion. The modifier is implemented with the Sparc ASI field. Again, Sparcle contains a number of new load/store instructions that differ only by the values that they place on the ASI pins during data cycles. These new load/store instructions are important to the implementation of full/empty bit synchronization and fast messages. The trap access signals are new versions of the Sparc memory exception signal MEXC, which have distinct trap vectors. These invoke context-switch and synchronization traps. The external condition bits are implemented through the Sparc coprocessor condition codes (CCC); consequently, "branch on condition-code" instructions in Sparc can be used to examine them.

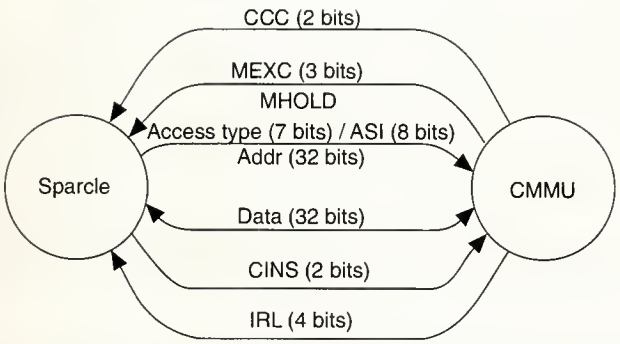


Figure 13. Sparcle signal names.

Finally, the external instruction interface is implemented directly through a Sparc coprocessor interface. Sparcle asserts one of the CINS signals to indicate that a coprocessor instruction has been decoded by the processor and should be executed by the coprocessor. Two CINS signals are required because pipeline interlocks can occasionally cause the instruction fetch unit to get ahead of the rest of the pipeline.

Latency tolerance. We already discussed rapid context switching for latency tolerance from the standpoint of the Sparcle processor. In addition to those Sparcle mechanisms, the cache controller must be able to handle multiple outstanding requests. This involves the ability to handle split-phase memory transactions (separating the request for data from the response) and to place returning data into the cache while the processor is performing some other task. Consequently, when the processor requests a data item that is not in the local cache, the cache controller asserts the appropriate trap line to initiate execution of the context-switch trap handler. At the same time, it sends a request message to the particular node that contains the requested data. Note that the mechanisms required to handle context switching differ little from those required for software prefetching. (However, see Kubiatowicz, Chaiken, and Agarwal[19] for some interesting forward-progress issues.)

Full/empty-bit synchronization. Full/empty-bit synchronization, as implemented in Alewife, requires support from the cache controller. Since full/empty-bit synchronization employs one synchronization bit for each data word, extra

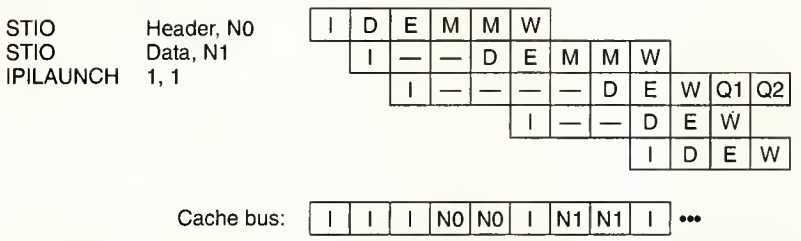


Figure 14. Pipelining for transmission of a message with a single data word.

storage must be reserved for these bits in the cache system. While these bits logically belong with the cache data, the Alewife CMMU implements them with the cache tags. This has a number of advantages. It eliminates a need for an odd number of bits in the physical memory used for cache data. It also makes access to the tags file much faster than access to the cache data, both because the tags file is smaller and because no chip crossings are required. This permits synchronization operations to occur in parallel with processing of the cache tags.

Of the Sparcle mechanisms, those important to full/empty synchronization are the external condition code, the access modifier (ASI), and one of the extra trap lines. All of the new synchronizing load/store instructions mentioned earlier are distinguished by the value of the ASI field that they generate (and whether they are read or write operations). For each data access, the Alewife CMMU takes the proffered ASI value along with the address and type of access. The CMMU uses the address to index into the tags file, retrieving both the tag and the appropriate full/empty bit. Simultaneously, it decodes the ASI value to produce two different actions, one which will be taken if the full/empty bit is full, and one if the full/empty bit is empty. When the tag lookup is completed, the CMMU completes both tags match and full/empty-bit operations simultaneously, either flagging a context-switch (on cache miss), a synchronization fault, or successful completion of the access. In all cases, the CMMU places the full/empty bit that was first retrieved from the tags file in one of the external condition codes for future examination by the processor.

The support that Alewife provides for full/empty-bit synchronization is external to the processor pipeline: that is, it occurs at the first-level cache. Consequently, full/empty bits never enter the processor core. Further, individual load/store instructions have varied semantics with respect to the full/empty bit: some cause test-and-set-like operations; others invoke traps. This places some data processing logic within the first-level cache. For modern processors that have one level of on-chip caching, a closer integration between the processor pipeline and full/empty bit synchronization might be desirable. This could include widening of internal processor registers and use of special full/empty-bit synchronization instructions that are sandwiched between Alpha-style[20] load-locked/store-conditional synchronization instructions.

Fast message handling. Fast messaging in Alewife relies on a number of features in the CMMU. All of the network queuing and DMA mechanisms are a part of this chip. Sparcle interfaces with these mechanisms through both the external instruction interface and through special loads and stores. As we discussed, Sparcle reserves one special load/store instruction (and corresponding ASI) for rapid descriptions of outgoing messages and rapid examination of incoming messages. The cache controller recognizes accesses with this ASI and causes data transfer to and from message queues instead of the cache. Message data thus transfers between the processor and network at the same speed as cached accesses.

Alewife uses the external instruction interface to implement the message launch mechanism. Consequently, message launches can be pipelined. Figure 14 gives a simple pipeline example. Here, the two-cycle latency for stores and the lack of an instruction cache limit the message throughput. More aggressive processor implementations would not suffer from this limitation. In this figure, Sparcle pipeline stages are Instruction fetch, decode, execute, memory, and writeback. Network messages are committed in the writeback stage. Stages Q1 and Q2 are network queuing cycles. The message data begins to appear in the network after stage Q2. Note that the use of DMA on message output adds additional cycles (not shown in the figure) to the network pipeline.

The close coupling between the message launch mechanism and the processor pipeline allows us to identify a precise launch completion point (corresponding to the writeback stage of the launch instruction). As a result, message launches are atomic. Before the launch instruction commits, no data is placed into the network. After the launch commits, Alewife sends a complete output packet to the network. These atomic semantics allow multiple levels of user and interrupt code to share a single network output port without requiring that the user disable interrupts before beginning to describe a message.

THE SPARCLE CHIP INCORPORATES MECHANISMS required for massively parallel systems in a Sparc RISC core. Coupled with a CMMU, Sparcle allows a fast, 14-cycle context switch, an 8-cycle user-level message send, and fine-grain full/empty-bit synchronization.

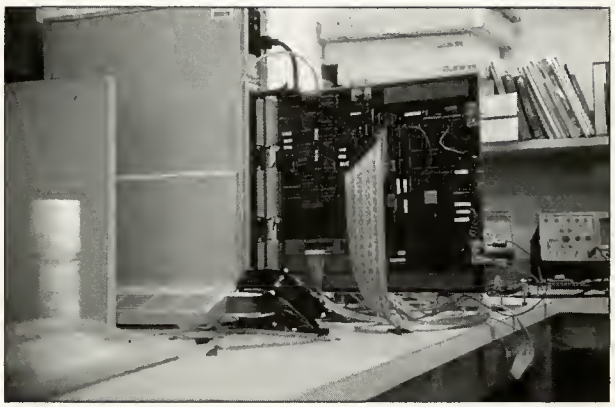


Figure 15. Sparcle's test system.

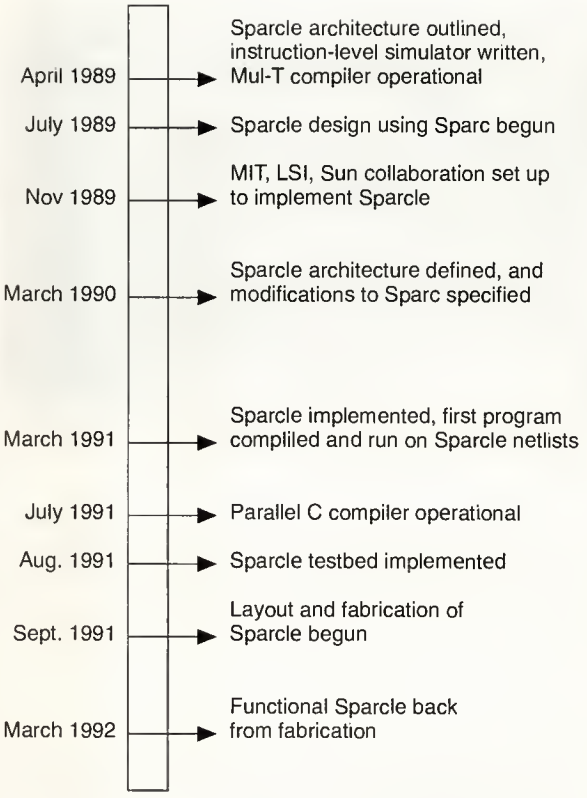


Figure 16. Sparcle's implementation schedule.

Before we received working Sparcle chips from LSI in the spring of 1992, we used an operating single-node test system. Also operational for several months was a compiler and a runtime system for our parallel versions of C and Lisp. The test system shown in Figure 15 comprises 256 Kbytes of static RAM memory, an I/O interface to the VMEbus for downloading programs and monitoring execution, and control logic to exercise the full/empty bit and context switching functionality. We had debugged the test system using Sparcs in place of Sparcles; it operated at a maximum clock frequency of about 25 MHz. (Sparc and Sparcle have only a few differing pins, and Sparcle even provides an input signal Mode pin that allows switching between Sparc and Sparcle modes.)

We have been running several parallel programs, including Sparcle's runtime system, to exercise all of Sparcle's functionality, at the maximum speed of the test bed. Scope measurements of critical signal timings on the chip's pins suggest we will be able to run the chips in an Alewife node board at roughly the same speed as the original, unmodified Sparcs.

Implementation of the Sparcle development relied on modifying an existing design through a unique collaboration with industry. Although we had our moments of trepidation, given the number of participants and the multiple failure modes (both technical and political), we believe this model of experimentation has been very successful. This implementation strategy not only allowed us, at a university, to experiment with architectural ideas in a real, contemporary processor design, it also significantly reduced the design effort from the concept stage to working chip.

Figure 16 depicts the resulting project schedule for Sparcle. We defined Sparcle's early architecture in April 1989. At MIT we also wrote a Sparcle compiler for a version of Lisp and implemented a cycle-by-cycle simulator. Later, we also developed a compiler for a parallel version of C. By March 1990, we had developed a detailed specification of the modifications to Sparc required to implement Sparcle. Then, Sun made high-level changes to Sparc functional blocks, and LSI made lower gate-level changes. We tested these changes against Sparcle binaries produced at MIT. Then LSI synthesized netlists and MIT tested them against several hundred thousands of test vectors. The test vectors included both Sparc vectors provided by LSI and Sparcle vectors obtained from the MIT Sparcle simulator. The test setup included a netlist module for the floating-point coprocessor and a behavioral model for the rest of the memory and communication systems. Finally, LSI undertook layout and fabrication, during which time we also implemented a test system for Sparcle.

While the Sparcle chip project demonstrates that a contemporary RISC microprocessor can readily incorporate features considered by many to be critical for massively parallel multiprocessing, the end systems benefit of these mechanisms can only be evaluated in the context of a complete multiprocessor system. We are in the final stages of implementing the Sparcle-based Alewife multiprocessor system. Figure 1 shows an Alewife node board with the Sparcle and FPU. Figure 17 shows a 16-node Alewife system package developed by the Advanced Production Technology group

Figure 17. The 16-node Alewife package.

at the Information Sciences Institute in Los Angeles. The CMMU chip has been implemented and tested. It is being implemented in LSI Logic's LEA 300K process, and we expect to begin its fabrication shortly.

Acknowledgments

The Sparcle project is funded in part by DARPA contract N00014-87-K-0825 and in part by National Science Foundation grant MIP-9012773. LSI Logic and Sun Microsystems helped implement Sparcle, and LSI Logic supported the fabrication of Sparcle. We acknowledge the contributions of Dan Nussbaum, who was partly responsible for the processor simulator and runtime system, and was the source of several ideas. Halstead's work on multithreaded processors influenced our design. Our research also benefited significantly from discussions with Bert Halstead, Tom Knight, Greg Papadopoulos, Juan Loaiza, Bill Dally, Steve Ward, Rishiyur Nikhil, Arvind, and John Hennessy.

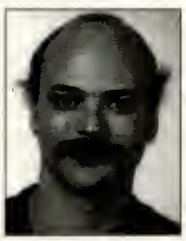
References

1. *Sparc Architecture Manual*, Sun Microsystems, Mountain View, Calif., 1988.
2. A. Agarwal et al., "The MIT Alewife Machine: A Large-Scale Distributed-Memory Multiprocessor," *Proc. Workshop on Scalable Shared Memory Multiprocessors*. Kluwer Academic Publishers, Boston, 1991. An extended version of this paper has been submitted for publication and appears as MIT/LCS Memo TM-454, 1991.
3. A. Agarwal et al., "APRIL: A Processor Architecture for Multiprocessing," *Proc. 17th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, Los Alamitos, Calif., June 1990, pp. 104-114.
4. B.J. Smith, "Architecture and Applications of the HEP Multiprocessor Computer System," *Soc. of Photooptical Instrumentation Engineers*, Bellingham, Wash., Vol. 298, 1981, pp 241-248.
5. Arvind, R.S. Nikhil, and K.K. Pingali, "I-Structures: Data Structures for Parallel Computing," *Trans. on Programming Languages and Systems*, Vol. 11, No. 4, Oct. 1989, ACM Press, pp. 598-632.
6. M. Dubois, C. Scheurich, and F.A. Briggs, "Synchronization, Coherence, and Event Ordering in Multiprocessors," *Computer*, Vol. 21, No. 2, Feb. 1988, pp. 9-21.
7. S. V. Adve and M.D. Hill. "Weak Ordering—A New Definition," *Proc. 17th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, June 1990, pp 2-14.
8. D. Lenoski et al., "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor," *Proc. 17th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, June 1990, pp. 148-159.
9. D. Kranz et al., "Integrating Message-Passing and Shared-Memory; Early Experience," to be published in *Conf. Principles and Practice of Parallel Programming*, ACM, May 1993, and appears as MIT/LCS Memo TM-478, 1993.
10. D. Chaiken, J. Kubiatowicz, and A. Agarwal, "LimitLESS Directories: A Scalable Cache Coherence Scheme," *Proc. Fourth Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS IV)*, ACM, Apr. 1991, pp. 224-234.
11. W.J. Dally et al. "The J-Machine: A Fine-Grain Concurrent Computer," *Proc. Int'l Federation for Information Processing (IFIP) 11th World Congress*, Elsevier Scientific Publishing, New York, 1989, pp. 1147-1153.
12. C.L. Seitz et al., "The Design of the Caltech Mosaic C Multicomputer," *Proc. 1993 Symp. Research on Integrated Systems*, G. Borriello and C. Ebeling, eds., MIT Press, Cambridge, Mass., 1993, pp. 1-22.
13. W-D. Weber and A. Gupta, "Exploring the Benefits of Multiple Hardware Contexts in a Multiprocessor Architecture: Preliminary Results," *Proc. 16th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, June 1989, pp. 273-280.
14. K. Kurihara, D. Chaiken, and A. Agarwal, "Latency Tolerance Through Multithreading in Large-Scale Multiprocessors," *Proc. Int'l Symp. Shared Memory Multiprocessing*, IPS Press, Japan, Apr. 1991, pp. 91-101.
15. G. Alverson, R. Alverson, and D. Callahan, "Exploiting Heterogeneous Parallelism on a Multithreaded Multiprocessor," *Workshop on Multithreaded Computers, Proc. Supercomputing*, ACM Sigraph, Nov. 1991.
16. G.M. Papadopoulos and D.E. Culler, "Monsoon: An Explicit Token-Store Architecture," *Proc. 17th Ann. Int'l Symp. Computer Architecture*, IEEE CS Press, June 1990, pp. 82-91.
17. D. Johnson, "Trap Architectures for Lisp Systems," *Proc. 1990 ACM Conf. on Lisp and Functional Programming*, 1990, pp. 79-96.
18. D. Ungar et al., "Architecture of SOAR: Smalltalk on a RISC," *Proc. 1984 Int'l Symp. on Computer Architecture*, 1984, pp. 188-197.
19. J. Kubiatowicz, D. Chaiken, and A. Agarwal, "Closing the Window

of Vulnerability in Multiphase Memory Transactions," *Fifth Int'l Conf. Architectural Support for Programming Languages and Operating Systems (ASPLOS V)*. ACM, Oct. 1992, pp. 274-284.
20. *Alpha Architecture Reference Manual*, Digital Press, Maynard, Mass., 1992.

Anant Agarwal serves at the Laboratory for Computer Science at MIT where he is an associate professor of electrical engineering and computer science. His current research interests include the design of scalable multiprocessor systems, VLSI processors, compilation and runtime technologies for parallel processing, and performance evaluation. At Stanford University, he participated in the MIPS and MIPS-X projects. He initiated the Alewife project at MIT, which is aimed at the design and implementation of a large-scale cache-coherent multiprocessor.

Agarwal received a B Tech in electrical engineering from the Indian Institute of Technology, Madras, India, and an MS and PhD in electrical engineering from Stanford. He is a member of the ACM and IEEE Computer Society.



John Kubiatowicz is a doctoral candidate in the Department of Electrical and Computer Science at MIT. His current research interests include parallel computer architecture, high-performance microprocessor design, and high-energy particle physics.

Kubiatowicz received BS degrees in electrical engineering and physics and an MS in electrical engineering from MIT.

David Kranz has been a research associate in the MIT Laboratory for Computer Science since 1987. His research interests are in programming language design and implementation for parallel computing.

Kranz received a BA from Swarthmore. While earning a PhD at Yale, he worked on high-performance compilers for Scheme and applicative languages.

Beng-Hong Lim is currently a doctoral candidate in MIT's Department of Electrical Engineering and Computer Science. His research interests include parallel computing and computer architecture. He received a BS and an MS in electrical engineering and computer science from MIT.

Donald Yeung is a PhD candidate at MIT. His research interests are in the area of multiprocessor design, including efficient hardware and software mechanisms for synchronization and latency tolerance.

Yeung received a BS from Stanford in computer systems engineering, and recently completed an MS in electrical engineering and computer science at MIT.

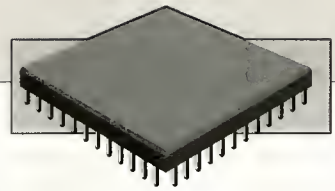
Godfrey D'Souza has been with the Sparc Systems Division and the CoreWare Group at LSI Logic where he is a senior design engineer involved with aspects of microprocessor and system level design.

D'Souza received a BS in electronics and communications engineering from the University of Baroda, India, and an MS in electrical engineering from the University of Washington, Seattle. He is a member of the IEEE Computer Society.

Mike Parkin is a staff engineer at Sun Microsystems, where he is currently part of a research team that is building a Sparc-based scalable multiprocessor system.

Parkin received a BSEE from Iowa State University and an MSEE from Stanford.

Direct questions concerning this article to David Kranz, MIT Laboratory for Computer Science, 545 Technology Square, Cambridge, MA 02139, or via e-mail at kranz@lcs.mit.edu.



Message-Routing Systems for Transputer-Based Multicomputers

An efficient message-routing system, an essential component of a multicomputer, allows communication between any two processes of a concurrent program wherever they are located on a parallel computer network. It simplifies the concurrent software development on multicomputers by separating the hardware architecture from the software configuration of processes. This article surveys some implemented routers for transputer networks and compares them for adaptivity, deadlock freedom, network latency, generality, and livelock freedom.

Domenico Talia

CRAI

Solving the communication problems inherent in a multicomputer composed of a large number of computing elements requires efficient data handling and interconnections. The processor in each computing element is tightly coupled to a local memory that is physically separated from the memories of other computing elements (Figure 1). Thus, on a multicomputer a message-routing system is necessary to control the passing of data between different processors. It lets us simplify the concurrent software development on parallel computers by separating the hardware architecture from the software configuration of processes.

One solution involves the single-chip Inmos transputer. This complete microcomputer contains communication links that interconnect to other transputers to form multicomputer systems. In particular, the T400 and T800 generations of transputers support communication among processes running on processors directly connected by a physical link. However, many parallel algorithms require a process running on one processor to communicate with processes running on other processors not directly connected by a link. To implement these algorithms on transputer-based multicomputers, we must include a through-routing system.

Recently, designers have introduced several message-routing systems for transputer-based multicomputers for direct use by a programmer.[1-6] In addition, some communication facilities have been provided in developing environments such as Express, Trollius, and Helios. Many of the routing systems are software implementations of routing algorithms that are defined for more general network topologies. Two of these, Interval Labelling[3] and Mad Postman,[5] will be implemented in hardware by a routing chip that connects to transputers and handles the message traffic.

Inmos is implementing the Interval Labelling routing in its IMS C104 communication device, which will enable communications among T9000 transputers. The MP1 network chip will incorporate the Mad Postman routing strategy proposed by Yantchev and Jesshope.[7] This device is not committed to interface only to the transputer; in fact, an interface is required between the processor that uses the MP1 chip to route messages. Two interfaces to the transputer have been developed. To date, neither the IMS C104 nor the MP1 is commercially available.

Another solution to communication problems on transputer networks is the link-switching network. In this approach, a set of switched links provides a highly connected, virtual network. Several link switch devices have been implemented and are currently in use. The best known is the Inmos C004 crossbar switch, which can switch

0272-1732/93/0600-0062$03.00 © 1993 IEEE

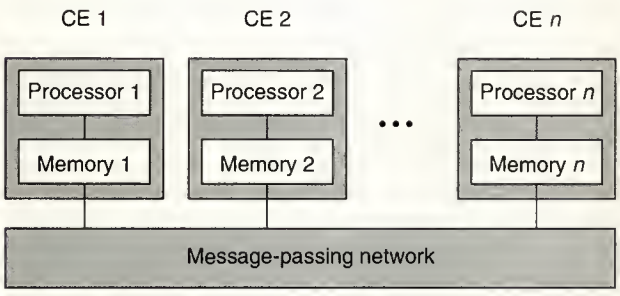


Figure 1. Multicomputer architecture.

32 input links to 32 output links with 20-Mbit/s operating speed. Other link switches support the Meiko Computing Surface, in the Floating-Point Systems T-series machines, and in other commercial transputer-based multicomputers.

In particular, the ESPRIT Supernode Project has implemented a link switch system for the Telmat T.Node system.[8] In it a pair of switch chips can switch 72 input links to 72 output links. Thus, four pairs of switch chips can switch 72 transputers into any link configuration.

With switched networks messages sent between two nodes must also pass through link switches, incurring an additional transmission delay. A user-required topology can be set on the network, but it is not a real alternative to through routing if dynamic link reconfiguration is not enabled. However, dynamic link reconfiguration requires an overhead due to the time taken for satisfying link switch requests.

Here, I survey some implemented routing systems for transputer networks and compare them with respect to several criteria, such as adaptivity, deadlock freedom, generality, livelock freedom, and network latency.

Except for one, these message-routing systems support the programming of concurrent applications on transputer networks. They present a user view of a transputer network as if all the transputers were completely connected. The message-routing systems are Tiny,[1] CSN,[4] Multiple Rings,[2] and Ordered Dimensions.[6] Finally, for its primary importance in future transputer-based multicomputers, I also describe the Interval Labelling routing that will be used in the latest generation T9000 transputer.

Message routing

As mentioned before, in a multicomputer system composed of a set of computing nodes connected by a communication network without shared memory, one of the main problems that must be faced is the routing of messages between the nodes. (See the Routing algorithms box.)

To be a practical communication support for programs running on multicomputers, a routing system must offer the following major features:

Routing algorithms

A routing algorithm determines a path between two nodes that are not directly connected. That is, it determines which output channel on a node i must route a message directed to a process running on a remote node j, as shown in Figure A. More formally, a routing algorithm is a *routing function* **R**: $N \times N \rightarrow C$ that maps the current node n_c and destination node n_d to the channel c_n on the route from n_c to n_d, $\mathbf{R}(n_c, n_d) = c_n$.

Routing algorithms can be divided into two main classes, *adaptive* and *oblivious*. In adaptive algorithms the routing paths are established dynamically, whereas in oblivious algorithms they are statically defined. *Deterministic* routing is a special case of oblivious algorithms in which a single route exists between two nodes. In fact, in deterministic algorithms, the route followed by a message sent from node i to node j is predetermined by its source-destination pair (i, j). For example, the communication systems of the Symult series 2010 and the Intel iPSC/2 employ an oblivious routing technique.

Another way to classify routing algorithms is based on the policy used to propagate a message from node to node. In *store-and-forward* routing a message is first entirely stored in each node on the path and then it is transmitted to the next node.

Different techniques are *cut-through* and *wormhole* routing. According to these techniques a message is broken down in *flits*, the smallest unit of information that a channel can accept or refuse. Instead of storing a message in a node and then transmitting it to the next node, the wormhole and cut-through techniques operate by advancing the head of a message directly from an incoming to an outgoing channel. Only a few flits are buffered at each node. The first flit (the *head*) holds the destination's address. Once a link is occupied by the head, it cannot be used for other messages until the last flit of the message has left it. With wormhole routing, blocked messages remain in the network. Virtual cut-through differs from wormhole routing in that it buffers messages when they block, removing them from the network.

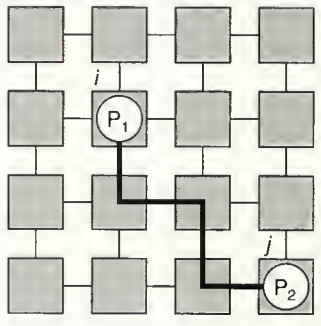


Figure A. A path between two processes.

Definitions

Adaptivity: The ability to choose the message route depending on traffic load and network topology. In adaptive algorithms the routing function is based on measurements or estimates of message traffic and network configuration.

Deadlock: A condition that occurs in a communication network when no message is buffered at its destination and no message can advance because all buffers in each routing path are full.

Deadlock freedom: The ability to avoid or recover from a deadlock occurrence.

Generality: The ability of a message-routing system to be used on one or more classes of network topologies.

Livelock freedom: A property that guarantees a message will be delivered to its destination in a finite time.

Network latency: The time it takes a message to leave the source and reach its destination.

- the routing system must be free of deadlocks;
- the message latency must be low; and
- no message may be infinitely delayed in the network.

Other relevant features are adaptivity and generality. (See Definitions box.)

Deadlock freedom. Deadlock is a typical problem of distributed computing. Since every distributed algorithm must solve a deadlock situation, to be practical, a message-routing system must be properly designed to avoid this occurrence.

Generally, in routing algorithms deadlock is related to the presence of cyclic paths in the network. Several techniques implement deadlock freedom in routing systems. For example, a simple policy may discard a preempted message. In this case the source node must be notified, and a retransmission mechanism must be provided.

An interesting technique to avoid deadlock is based on an ordering of the network channels.[9] Physical channels belonging to cycles are split into a group of virtual channels. The virtual channels are ordered, and message routing is restricted to visit channels in decreasing order to eliminate cycles.

Network latency. In store-and-forward routing the network latency (T_l) is the product of the mean internode latency (hop time) and the average distance (number of hops):

$$T_l = T_{l1}\, N_h$$

where T_{l1} is the hop time and N_h is the number of hops. In wormhole routing the network latency is

$$T_l = T_{h1}\, N_h + (K/B)$$

where T_{h1} is the routing delay in each node for sending the message head to its destination. N_h is the number of hops, and K/B is the time required to move the whole message K (bytes) through the wormhole channel of bandwidth B (bytes/s).

Since high communication costs may abate the benefits deriving from the parallel execution of programs, a practical routing system should have low mean latency to deliver messages to their destinations.

Livelock freedom. Livelock occurs when a message never arrives at its destination. Oblivious routing avoids livelock if the queue buffering is fair and the paths are minimal. In adaptive routing this is not sufficient; generally, a priority must be assigned to each message when it is sent. This priority will be increased as the message remains in the network, and messages are routed respecting their priority.

Adaptivity. Although an adaptive algorithm may be restricted to a particular topology, it might achieve a better performance than an oblivious one. The system can adapt to traffic conditions and choose the communication paths, avoiding hot spots. Moreover, adaptive algorithms help programmers increase their productivity because they do not need to analyze network hot spots at coding time.

Generality. While some routing systems work in one topology class (such as hypercubes, meshes, trees), some other systems support a particular network. A routing system for transputer-based multicomputers can be considered as general if it runs on the topologies that can be built using the four physical links: mesh, ring, tree.

Routing systems

The five routing systems for transputer networks are Tiny, Multiple Rings, CSN, Ordered Dimensions, and Interval Labelling. Except for Interval Labelling, which will be used with T9000 transputers, these systems actually support through-routing in concurrent applications implemented on transputer-based multicomputers.

Tiny. Tiny is a message-routing harness developed for T800 transputers at Edinburgh Parallel Computing Centre.[1] In Tiny the message routing is based on routing tables recording the paths between any two processors in the network. At starting time Tiny explores the network topology and builds routing tables, storing the shortest paths (one or more) from each processor to each other.

Tiny implements two point-to-point routing strategies, sequential and adaptive, and a broadcast (one-to-every-other) strategy. Users may select the two point-to-point routing strategies at initialization time. Every strategy uses the store-and-forward technique to propagate messages.

The sequential strategy is implemented by using the same shortest path from any through-routing processor toward the destination. This strategy provides provable deadlock free-

dom by eliminating cycles in routing tables when these are built. This is achieved by routing a message to a destination through different links on a processor, depending on the link through which that message reached that processor. A useful feature of the sequential strategy is that it guarantees messages will arrive at their destination in the order in which they were sent.

The adaptive strategy determines at each through-routing processor which of the shortest paths from that processor to the destination is the best. To make the decision, Tiny examines the output queues of the local processor's appropriate links and enqueues the message to the link with the shortest queue.

To implement broadcast communications, Tiny uses a third strategy. A message broadcast by a process is received directly by all the other processes in the network instead of being restricted to only one process. The broadcast strategy uses a broadcast tree determined during initialization for each processor P_i. Information is stored in every other processor P_j to indicate its relative position in the P_i tree. Figure 2 shows a broadcast tree from node P_1.

Tiny is a communication kernel running on each transputer and connected to one or more client processes by channels. Several agents (processes implemented in C language and transputer assembler) on each transputer implement Tiny. Each process handles an external link or a channel connecting Tiny to a user process. Tiny offers to a user an interface that provides a set of communication primitives implementing read and write operations from and to any other process in the network:

```
void pktRead (CHAN *in, int *source, *message, size)
void pktWrite (CHAN *out, int *dest, *message, size).
```

Tiny also provides a broadcast communication function:

```
void pktBroadcast (CHAN *out, int *message, size).
```

Some experiments have measured Tiny's performance in routing messages on a network of T800 transputers with 20 Mbits/s links. The internode latency (one hop) for 64-byte messages is about 200 microseconds and for 256-byte messages about 500 µs with a low load in the network.

Multiple Rings. CRAI (Consorzio per la Ricerca e le Applicazioni di Informatica)[2] designed and implemented this routing system. It is based on a deadlock-free adaptive routing algorithm for *k*-ary *n*-cubes, which extends Roscoe's proposal for ring topologies.[10] The transputer implementation represents a two-dimensional version of the algorithm. Four defined interconnected rings may transfer messages between rings according to the shortest path and the network load.

Messages transfer from node to node using the store-and-forward technique. In this routing system deadlock is avoided, assuring that on each ring a node outputting a message must be prepared to accept a message from that ring. To allow rerouting from one ring to another, this system always maintains one free buffer for input for each ring when ring input is required. Thus, the buffer space required by the router to guarantee deadlock freedom is very small, and it is independent from the network dimension (that is, number of nodes).

Multiple Rings uses four rings connecting all the transputers that compose the two-dimensional torus. Two of the rings run along the columns and two along the rows, with the message traveling in opposite directions on each of the row and column rings (Figure 3). The design of the routing system incorporates these criteria:

- after a message has been output on a ring, the node will be prepared to receive an input message on that ring;
- a message can be transferred between rings only if the node has adequate buffer space; and
- the binding of a message to a node's output link need not occur until that output link becomes available.

Multiple Rings is adaptive. Once a message is input from a ring, the system generates three possible rings on which the message might be output: PreferredRing, OptionalRing, and RequiredRing. The first two rings optimally reduce the Cartesian distance; the third one is the ring on which the message is received. If possible, the routing occurs by choosing the ring for output that

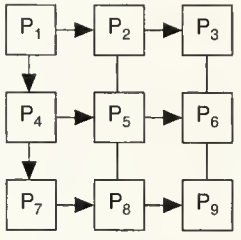


Figure 2. A broadcast tree from P_1.

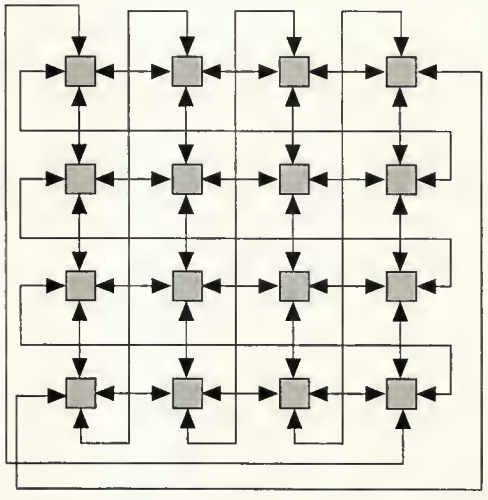


Figure 3.Two-dimensional toroidal topology.

Multiple Rings applies to all of the computer networks in which at least one Hamiltonian circuit can be identified.

reduces the Cartesian x,y mesh distance from the current node to the node to which the message is addressed. This routing does not guarantee that the message will ultimately arrive at its destination. If the message fails to arrive at its destination within some interval, routing relative to the rings, not Cartesian coordinates, begins.

To identify when the routing should change, each message contains an initial jump count. When the message's jump count reaches zero, the message transfers between rings, only to reduce the ring distance to the target node. If the message can transfer to a ring that reduces the distance to the destination, the message will reach the target node on this ring. Otherwise the message will flow on the original ring until reaching the target node. This approach guarantees livelock freedom. It also tends to reduce the local congestion because some of the messages will be routed with a different strategy.

The system has been implemented in the Occam 2 language. The algorithm is scalable in terms of managed rings and then can exploit the overall network bandwidth. Note that this routing system applies to all of the computer networks in which at least one Hamiltonian circuit can be identified.

One source presents a performance study of the system for a network of T800 transputers with 10-Mbit/s links.[2] Other experiments used links with a 20-Mbit/s bandwidth. In this case the mean internode latency increases with demand from 250 to 600 µs for 64-byte messages and from 450 to 1,200 µs for 256-byte messages.

CSN. The Computing Surface Network communication system developed at Meiko Scientific Ltd.[4] provides full through-routing support for concurrent programs running on the Meiko Computing Surface.

The Computing Surface is a multiuser, parallel computer based on a mix of T800 transputers, Sparcs, and i860 processors. In fact a single large Computing Surface can be viewed as a number of independent machines defining a number of user domains. Each such domain contains a set of processing elements that are accessible to the user. A generic processing element has four main components: compute processor, memory system, communication engine, and control interface. The communication engine comprises one or more transputers, which implement the CSN system.

CSN implements communication between two or more processes by means of an interconnection entity called a transport. It is an asymmetrical and bidirectional way of communication that has one owner process. A transport is a service point to which a process may send a message and from which it may receive a message without specifying the source.

The basic idea of this communication system is similar to that of other conventional communication mechanisms, such as Unix Streams. Message exchange among processes is not carried out through channels, as in the other routing systems discussed here, but by means of network access points, the transports, which might be used by many processes. A transport can be created dynamically from a process, and after its creation it can be used in each direction (send or receive) for data communication with a dynamic set of partners.

For transport management, CSN provides a distributed name service called directory server. The directory server dynamically associates a network physical address to each transport name. When a transport is created, its identifier includes three fields:

- *A global name.* This unique name is defined by the owner and must be used by partners.
- *A local name.* This is the name of the transport used in the owner process to send and receive through that transport.
- *A network address.* This is a physical address associated with the transport in the processor network. To this address is associated a buffer on which all the messages exchanged using that transport are enqueued.

When a process tries to receive a message, it does not need to specify the name of the sender; it is sufficient to use the local name of the transport on which the message will be read. Two primitives are provided to send and receive data:

csn_tx(Trans, ..., Msg) and
csn_rx(Trans, ..., Msg).

Trans is the transport name; Msg is the message to be exchanged. The communications can be synchronous or asynchronous and specify a different parameter in the primitives.

As in the other routing systems, data exchanges through a network of processes located on each transputer of the Computing Surface. For processor-to-processor message transmission, CSN uses a technique similar to virtual cut-through, dividing messages into units of 32 bytes. Even if a message smaller than 32 bytes is sent from an application's process, a 32-byte packet will travel on CSN.

Ciampolini, Corradi, and Leonardi[11] describe some experimental experiences with transports and present some measurement of message latency on a Meiko Computing Surface. The experiment used nine transputers connected in a mesh

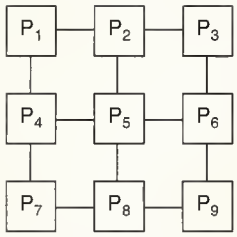


Figure 4. Two-dimensional mesh network.

topology and eight processes; each process ran on a different transputer and sent messages toward a receiver located on the ninth transputer. The latency to send 10,000 medium-size messages ranges from 10 to 60 seconds, depending on the transmission frequency. Taking into account that for this network the mean distance is 2.25, we obtain an internode latency ranging from 450 to 2,600 μs for medium-size messages.

Ordered Dimensions. The Inmos Central Application Group in Colorado Springs, Colorado, developed this message-routing system,[6] basing it on the ordered dimensions approach proposed by Dally and Seitz.[9]

This approach partitions the network channels into classes and orders them within classes in a way that reflects the topology of the network. Deadlock freedom derives from the fact that the channel structure does not contain cycles. Cycles are broken by adding virtual channels to the network. Several logically independent virtual channels might be mapped on a physical link.

Channel classes partition the unordered set of channels in a network. Each class C is an ordered set of channels. An ordered set of channels $C = (c_1, c_2, \ldots, c_n)$ defines a channel sequence that a message can follow from the source to its destination ($c_1 < c_2 < \ldots < c_n$). Classes are themselves ordered to impose a dependency on one another. Thus a traveling message having traversed the channels of one class will not revisit them after traveling through channels of the next class. Classes often correspond to dimensions of the network, for example, the x and y dimensions of a two-dimensional mesh.

As an example, the routing algorithm for a two-dimensional mesh (Figure 4) first defines two ordered dimension classes: the Y dimension consisting of north and south channels and the X dimension consisting of east and west channels. A message must complete its routing in the X dimension before traveling in the Y dimension. Each dimension class further consists of two independent direction classes. A message travels east or west, but never in both directions. Channels are ordered within each direction class.

Buffers are provided to each channel class. Additional internal channels allow switching from the higher order X dimension to the lower order Y dimension. The resulting logical network is shown in Figure 5.

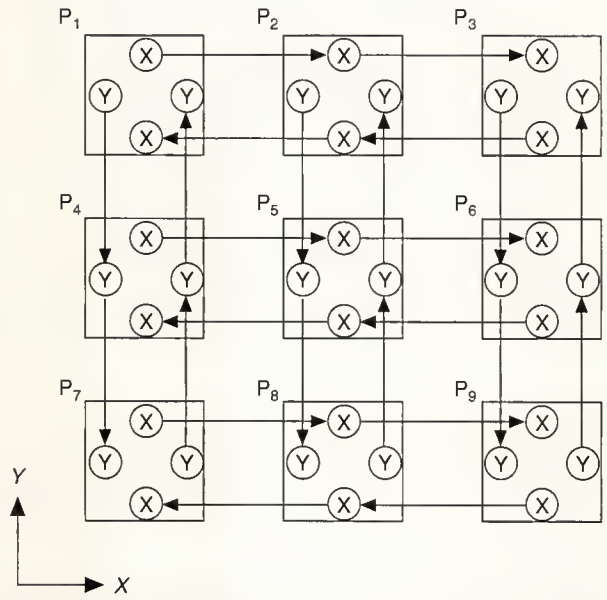


Figure 5. Logical network.

To route messages in this network, the algorithm compares the current Cartesian coordinates with that of the destination. The message will move along the east/west direction until the correspondent coordinate matches the destination. Then the algorithm switches to the Y dimension (north/south direction), and the procedure repeats until the destination is reached.

This routing system has been implemented for a three-dimensional hypercube and for a two-dimensional mesh of transputers. For node-to-node message transmission the system uses the store-and-forward technique. Shumway[6] describes the Occam 2 source code of the two routing algorithms.

Ordered Dimensions routing can be applied to any regular topology such as hypercubes, tori, meshes, and rings. The message routing is oblivious. In fact, the path taken by a message is statically defined because it is derived from the channel dependency graph. Thus a network cannot adapt its routing to the message traffic. Furthermore, the ordering over all the channels allows only one path between two nodes. Although this condition can lead the network to congestion, it may somewhat benefit a user because for each pair of nodes, messages are received in the order they were sent.

Another critical aspect of this system is that livelock freedom is not implemented. Guaranteeing livelock freedom requires provision of a fair scheduling of competing channels. However, the system does not use this kind of scheduling

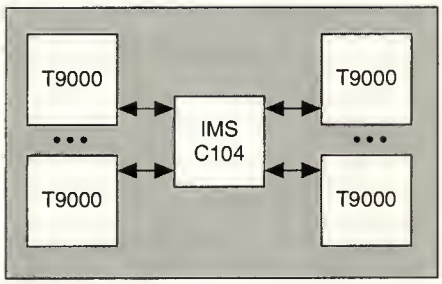


Figure 6. T9000 and IMS C104 connection.

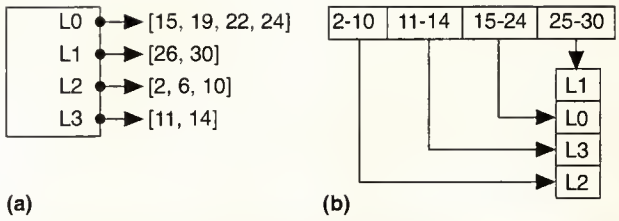


Figure 7. Interval Labelling: destinations (a) and table (b).

because it actually reduces the system performance.

The internode latency of the system on a T800 mesh topology increases with demand from about 100 to 300 μs with a message size of 64 bytes, and from 210 to 700 μs with a message size of 256 bytes.

Interval Labelling. Inmos is implementing this routing scheme in hardware. The IMS C104 communication support device for the T9000 transputer includes a full 32 × 32 crossbar switch, enabling messages to be routed from any of its links to any other link. One or more T9000s might be connected to a IMS C104 device, which enables the connection with other transputers in the network (Figure 6). In particular, a single C104 can provide full connectivity between 32 T9000s.

The Inmos T9000 is the latest generation transputer. The main goal for the T9000 was to improve the performance of the transputer while maintaining compatibility with existing transputer products. The T9000 provides an order-of-magnitude increase in performance with respect to the T800 and implements the same instruction set as the existing T805 transputer. In the T9000 transputer, designers used CMOS technology to integrate a 64-bit floating-point processor, a 32-bit integer processor, 16 Kbytes of cache memory, a virtual channel processor, and four high-bandwidth links on a single chip.

The key features of the T9000 architecture are a pipelined superscalar CPU combined with 16-Kbyte on-chip RAM and improved communications (80 Mbyte/s bidirectional bandwidth) to make multiprocessor programming easier. The pipelined superscalar architecture executes several instructions on each clock cycle and operates at a clock speed of 50 MHz. Instructions execute in a five-stage pipeline. The first stage can fetch two local variables; the second can execute two address calculations for accessing variables; the third can load two nonlocal variables; the fourth can perform an FPU or ALU operation; and the final stage can perform a write or conditional jump. The T9000 also provides a grouper to assemble groups of instructions. One group can be sent through the pipeline every cycle to make the best use of the hardware.

The major goal of achieving a significant performance increase produced a design with a peak performance in excess of 200 MIPS and 25 Mflops and a sustained performance exceeding 80 MIPS and 10 Mflops.

The communication facilities of the T9000 have been extended in comparison to the T800. On previous transputers the user was limited to assigning two software channels, one in each direction, to each hardware link. In the T9000 new multiplexing hardware allows the mapping of any number of channels on each physical link between two directly connected transputers. A communication processor called the virtual channel processor (VCP) handles these software channels. Whereas the VCP allows communication between two directly connected T9000 transputers, the C104 chip provides hardware support for routing messages across a network of T9000s.

With the T9000 and the IMS C104, we can define communication channels between two processes independently of where they are physically located or whether the channels are routed through the network. Each link of this network should support a bidirectional bandwidth of about 150 Mbps.

The routing algorithm used in the IMS C104 is called Interval Labelling.[12] This scheme assigns a distinct label to each transputer in a network. On each IMS C104, each output link has an associated interval (set of consecutive labels). The intervals associated with the links are not overlapping. As a message arrives, the algorithm examines the address to determine which interval contains a matching label then forwards the message along the associated output link.

Figure 7 shows an example of Interval Labelling in which a set of reachable destinations is defined for each link. According to the table, a message with destination address 22 ($15 < 22 < 24$) is routed through link 0 (L0).

Any network topology can be labeled so that the routing is deadlock free. This sometimes produces a nonoptimal routing that cannot exploit all of the links in the network, such as for ring topologies. On the other hand, optimal deadlock-free labelling is possible for trees, meshes, hypercubes, and multistage networks. These labellings will be provided with the routing system.

As an example, we show how to label a network with a

mesh topology. An N-dimensional mesh is composed of M meshes of dimension $N-1$, with M corresponding nodes, one from each $N-1$-dimensional mesh, joined to form a line. If each of the $N-1$-dimensional meshes has P nodes, these are numbered $0, \ldots, P-1; P, \ldots, 2P-1; \ldots; (M-1)P, \ldots, MP-1$. In this way a mesh may be labeled to route messages in a deadlock-free manner.

Figure 8 shows an example of a simple network composed of four T9000 transputers and three C104s. The interval notation [2,4) should be read as meaning that the label value must be greater than or equal to 2 and less than 4 to lie within this interval. If a message with label 3 is sent from T0, it passes through the three C104s before going to the T3 transputer.

Interval Labelling uses a wormhole technique in which the routing decision occurs as soon as the head of the message has been received. With a network of IMS C104s wormhole routing does not affect through-routing transputers, minimizing the network latency in the message transmission. Moreover, this routing system ensures livelock freedom.

The Interval Labelling routing is oblivious; in fact, it is not able to adapt the message routing according to the communication load of the network. This aspect may create an excessive amount of communication on some link that will become a hot spot in the network. To eliminate hot spots, the IMS C104 should optionally implement a two-phase routing algorithm in which each message is first sent to a random intermediate destination, then on to its final destination.

To implement this two-phase strategy, the system must prepend each message with a random header that indicates the intermediate address. To be sure that deadlock does not occur, the two phases must use separate links. This is possible by assigning random headers and destination headers from distinct intervals. Thus the interval associated with a given link on an IMS C104 must be a subinterval of the random or destination header range. Deadlock freedom is also assured. However, the two-phase routing algorithm introduces an additional communication overhead.

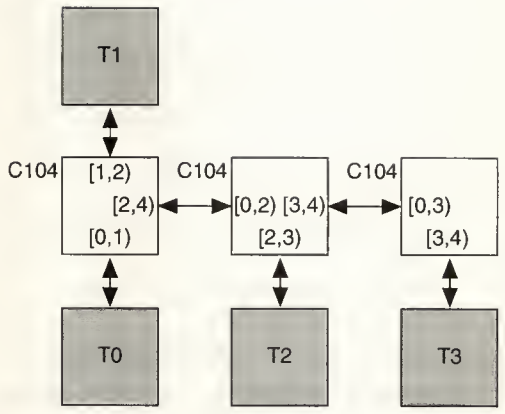


Figure 8. Interval Labelling of a simple network.

No experimental measures were available when this article was written. Some simulation results give the mean internode latency at about 2 or 3 μs. If these results are confirmed in the real use of the IMS C104 chip, the Interval Labelling routing system will represent a very good solution for message routing in the next generation of transputer-based multicomputers.

Comparisons

The message-routing systems discussed here can be effectively used to support network communications in parallel programs running on multi-transputer systems. They have different features, which can be compared for a better evaluation. Table 1 summarizes the main features of the routing systems.

Table 1. Summary of main features of the routing systems.

Systems	Transmission	Adaptivity	Deadlock freedom	Livelock freedom	Generality
Tiny	Store-and-forward	Yes No	No Yes	No Yes	All
Multiple Rings	Store-and-forward	Yes	Yes	Yes	All topologies with at least one Ham. circuit
CSN	Virtual cut-through	No	Yes	Yes	All
Ordered Dimensions	Store-and-forward	No	Yes	No	Any regular topology
Interval Labelling	Wormhole	No	Yes	Yes	All

Generally, the routing systems discussed here outperform the message-passing systems provided with the operating systems developed for transputer-based machines.

These routing systems are not used as part of an operating system. Parallel programs written using these systems can be linked and will work without changes on many hardware topologies. There is no unanimous answer to whether the message-routing facility should be in the operating system or, as in the systems discussed here, available without any higher abstraction level. While the integration in an operating system, such as Helios and Trollius, has its advantages—both because they offer a standard interface and often high-level primitives—it imposes high software overheads. Since the communication overhead is a very critical issue for most parallel programs running on multicomputers, many users prefer to use a message-routing system.

Deadlock freedom. As mentioned before, a system unable to avoid deadlock occurrence is not practical. All the routing systems surveyed here are deadlock free. The only exception is represented by the adaptive strategy that can be optionally used in Tiny. This routing strategy is not practical because cycles can be created in a message route so deadlock can occur.

The routing systems use different techniques to avoid deadlock. The approach used in CSN and Ordered Dimensions is to multiplex an acyclic virtual network across the communication links of a cyclic network. Multiple Rings avoids deadlock, assuring that on each ring a node that has output a message must be prepared to accept a message from that ring. To allow rerouting from one ring to another, this system always maintains a free buffer for input for each ring when ring input is required. Finally, in the Tiny and Interval Labelling systems certain link-to-link connections are not allowed to avoid cyclic message routes.

Livelock freedom. In the adaptive strategy used in Tiny, livelock freedom is not implemented. It is also not implemented in the Ordered Dimensions system in which livelock freedom introduces a large overhead. On the contrary, in the other routing systems no livelock may arise. In oblivious systems livelock is avoided if the queue buffering is fair, though in adaptive systems this is not sufficient. In fact, to assure livelock freedom in Multiple Rings when the message fails to arrive at its destination within some interval, the routing relative to the rings, not Cartesian coordinates, is performed.

Adaptivity. Multiple Rings uses a strategy that adapts the message routing depending on the network load (the adaptive strategy of Tiny is not of much practical use). In contrast, CSN, Ordered Dimensions, Tiny/sequential, and Interval Labelling use oblivious routing. Thus, these systems are unable to avoid network congestion when the communication load in a network is unbalanced.

To show the relevance of this issue, look at the iPSC/2, which employs an oblivious routing system. The message latency increases almost linearly as the number of messages that simultaneously reach a common node.[13] Adaptive systems, such as Multiple Rings, avoid this. Finally, note that although the Interval Labelling routing is oblivious, it should also implement the two-phase routing strategy that allows the elimination of network hot spots, although it adds some overhead.

Message transmission. Since the T800 generation of transputers does not implement direct link switching, messages must be entirely buffered at each intermediate node. This store-and-forward technique is implemented in Tiny, Multiple Rings, and Ordered Dimensions. CSN uses the more CPU-intensive virtual cut-through technique.

Interval Labelling uses wormhole routing. This method is very effective because the IMS C104 chips implement the routing in hardware. The message header traversing a network of IMS C104s creates a circuit through which the message flows. Thus a message can be passing through several IMS C104s at the same time without disturbing the processing transputers. This method minimizes the latency and separates communication from computation.

Network latency. Network latency is a major parameter to be evaluated in a routing system because high latency overhead may abate the benefits of parallelism in communication-intensive programs. Although on the basis of the measurements mentioned there is not a large difference among the internode latencies of the routing systems (the Interval Labelling cannot be compared), Ordered Dimensions outperforms the others. On the other hand, this system is not adaptive. This means that under a nonuniform traffic distribution the system performance will decrease dramatically, and the network may collapse.

Generally, the routing systems discussed here outperform the message-passing systems provided with the operating systems developed for transputer-based machines (Helios, Trollius). As mentioned before, the overhead of using a message-routing system hidden in an operating system is not small. For example, in Helios[11] the read and write system library calls provide a communication system. Using these calls results in internode latencies for 64-byte and 256-byte

messages of 1,150 and 1,300 µs, if the test nodes run only the processes performing the measurements (that is, no load).

When compared with the communication latency obtained for other multicomputers, such as the iPSC/2 and Ncube/10, the network latency of the message-routing systems presented here show their effectiveness. Two reports[14, 15] show the experimental internode latencies of the communication systems of the iPSC/2 and Ncube/10. With a message size of 64 bytes and very low demand, they are respectively 400 and 600 µs. With 256-byte messages, the internode latency is 780 µs for the iPSC/2 and 1,060 µs for the Ncube/10. With the same message sizes the systems presented here offer similar or better performance.

This comparison shows that these routing systems for transputer-based multicomputers are more than comparable with the hardware-implemented message-passing system of the Intel iPSC/2, and much more effective than the software-implemented communication system of the Ncube/10.

Generality. Generality is an important feature to promote a large use of a message-routing system. All the message-routing systems presented here can be used on a great variety of hardware topologies. In particular, Tiny, CSN, and Interval Labelling base their routing on tables, so they can be used on any network topology that can be set up by the four transputer links. Multiple Rings can be applied to all of the computer networks where at least one Hamiltonian circuit can be identified (ring, mesh, torus, etc.). The Ordered Dimensions system can be used in any regular network topology.

ONE OF THE MAIN PROBLEMS that must be faced in achieving a broad use of parallel computers is the lack of tools that hide the physical architecture and offer a high-level interface to a user. The routing systems presented here implement a virtual level that makes machine topology irrelevant from the programmer's point of view.

These five routing systems for transputer networks include four in use in current implementations that allow data exchange between processes mapped on transputers not directly connected. They are efficient software implementations of routing algorithms that can be used on a great variety of hardware topologies.

The fifth system (Interval Labelling) will be the routing system of the latest generation T9000 transputer. If the simulation results are confirmed in the real use of the IMS C104 chip, the Interval Labelling routing system will represent a very good solution for message routing in the next generation of transputer-based multicomputers.

Next-generation parallel computers will be multiuser general-purpose machines. To achieve this goal, efficient routing systems are needed. Therefore, message-routing systems will represent a critical aspect in the definition of a general-purpose parallel machine based on transputers or other processors.

On the other hand, the use of a general-purpose routing system allows the programmer to develop parallel programs without worrying about the details of the hardware configuration. This increases code portability and reusability. The approaches discussed here are in accordance with this technological trend that will bring many benefits to the implementation of real applications on multicomputer systems.

Acknowledgments

The CNR Progetto Finalizzato Sistemi Informatici e Calcolo Parallelo (Program on Information Systems and Parallel Computing) partially supported this work under grant 90.0076.69. A preliminary version of this article appeared in the *Proceedings of the WoTUG 15th Technical Meeting* published in 1992 by IOS Press, Amsterdam.

References

1. L. Clarke and G. Wilson, "Tiny: An Efficient Routing Harness for the Inmos Transputer," *Concurrency: Practice and Experience*, Vol. 3, No. 3, John Wiley & Sons, New York, June 1991, pp. 221-245.
2. M. Cannataro et al., "Design, Implementation and Evaluation of a Deadlock-Free Routing Algorithm for Concurrent Computers," *Concurrency: Practice and Experience*, Vol. 4, No. 2, John Wiley & Sons, Apr. 1992, pp. 143-161.
3. D. May, "Transputers and Routers: Components for Concurrent Machines," *Proc. Third Transputer/Occam Int'l Conf.*, IOS Press, Japan, May 1990.
4. Meiko Scientific Limited, *CSTools User Guide*, Bristol, UK, 1990.
5. P.R. Miller, C.R. Jesshope, and J.T. Yantchev, "The Mad-Postman Network Chip," *Proc. Transputing*, IOS Press, Amsterdam, 1991, pp. 517-536.
6. M. Shumway, "Deadlock-Free Packet Networks," *Proc. Second North American Transputer Conf.*, IOS Press, Amsterdam, Oct. 1989, pp. 139-177.
7. J.T. Yantchev and C.R. Jesshope, "Adaptive Low-Latency Deadlock-Free Packet Routing for Network of Processors," *IEE Proc.*, Vol. 136, No. 3, IEE, 1989, pp. 178-186.
8. D.A. Nicole, E.K. Lloyd, and J.S. Ward, "Switching Networks for Transputer Links," *Proc. Ninth OUG Tech. Meeting*, IOS Press, Amsterdam, 1988, pp. 147-165.
9. W.J. Dally and C.L. Seitz, "Deadlock-Free Message Routing in Multiprocessor Interconnection Networks," *IEEE Trans. Computers*,

Vol. C-36, May 1987, pp. 547-553.
10. A.W. Roscoe, "Routing Messages Through Networks: An Exercise in Deadlock Avoidance," *Parallel Programming of Transputer-Based Machines, Proc. Seventh OUG Tech. Meeting*, IOS Press, Amsterdam, 1988, pp. 55-79.
11. A. Ciampolini, A. Corradi, and L. Leonardi, "Testing Parallel Objects Kernel Environment (POKE)," Tech. Report Progetto Finalizzato Sist. Inf. e Calcolo Parallelo, No. 3/75, C.N.R., July 1991 (in Italian).
12. J. van Leeuven and R.B. Tan, "Interval Routing," *The Computer J.*, Vol. 30, No. 4, 1988, pp. 298-307.
13. O. Frieder et al., "Experimentation with Hypercube Database Engines," *IEEE Micro*, Vol. 12, No. 1, Feb. 1992, pp. 42-56.
14. J. Powell, "Performance Issues," *Helios 1.2 User Manual*, Perihelion Software Ltd., Nov. 1990.
15. X. Zhang, "System Effects of Interprocessor Communication Latency in Multicomputers," *IEEE Micro*, Vol. 11, No. 2, Apr. 1991, pp. 12-15, 52-55; correction June 1991, p. 6.

Additional readings

Close, P., "The iPSC/2 Node Architecture," *Proc. Third Conf. Hypercube Concurrent Computers and Applications,* SIAM, Philadelphia, Penn., Jan. 1988.

Debbage, M., M.B. Hill, and D.A. Nicole, "Virtual Channel Router Version 2.0 User Guide," Tech. Report, Dept. of Electronics and Computer Science, Univ. of Southampton, UK, June 1991.

De Carlini, U., and U. Villano, "The Routing Problem in Transputer-Based Parallel Systems," *Microprocessors & Microsystems,* Vol. 15, No. 1, 1991, pp. 21-33.

Felperin, S.A., et al., "Routing Techniques for Massively Parallel Communication," *Proc. IEEE*, Vol. 79, No. 4, 1991, pp. 488-503.

Fielding, D.L., et al., "The Trollius Programming Environment for Multicomputers," *Proc. Third Conf. NATUG*, IOS Press, Amsterdam, 1990, pp. 207-210.

Garnett, N.H., "HELIOS—An Operating System for the Transputer," *Parallel Programming of Transputer Based Machines, Proc. Seventh OUG Tech. Meeting,* IOS Press, 1988, pp. 411-419.

Hu, L.R., and G.S. Stiles, "Fluid Dynamics on EXPRESS: An Evaluation of a Topology-Independent Parallel Programming Environment," *Proc. Fourth Conf. NATUG*, IOS Press, Oct. 1990.

Inmos, *The T9000 Transputer Products Overview Manual*, SGS-Thomson Microsystems, 1991.

Knowles, A., and T. Kantchev, "Message Passing in a Transputer System," *Microprocessors & Microsystems*, Vol. 13, No. 2, 1989, pp. 113-123.

Pancake, C. M., "Software Support for Parallel Computing: Where Are We Headed," *Commun. ACM*, Vol. 34, No. 11, 1991, pp. 53-64.

Surridge, M.W., "ECCL: A General Communication Harness and Configuration Language," *Proc. Second Int'l Conf. Applications of Transputers*, IOS Press, 1990.

Syscon Corporation, "VCX User Guide," Columbia, Md., 1990.

Tanenbaum, A.S., *Computer Networks*, Prentice-Hall, Englewood Cliffs, N.J., 2nd ed., 1989.

Domenico Talia is currently a member of the CRAI research team in the CNR-PFI project. He works in the area of parallel and distributed systems and is a senior researcher fellow of the CRAI parallel computing team. His technical interests include distributed systems, parallel architectures, and concurrent programming languages.

Talia studied physics at the University of Calabria, Italy. He is a member of the IEEE Computer Society, the Association for Computing Machinery, and AICA (the Italian Association of Informatics and Automatic Computing).

Address any questions concerning this article to the author at CRAI, Localita S. Stefano, 87036 Rende, CS, Italy; dot@crai.it.

Get to market faster with FPGAs

Ware Myers

Contributing Editor

Application-specific integrated circuits (ASICs) and mask-programmable gate arrays take time, often measured in months, to procure from a vendor. A field-programmable gate array (FPGA), in contrast, can be incorporated in a product within a few days after logic designers complete their work.

The time saved gets the product to market that much sooner. Moreover, if system test reveals a flaw, designers can correct it in another day or two. Later, if use of the product by early customers discloses a misunderstanding of the requirements, that, too, can be quickly corrected.

That fast implementation sounds promising. In fact, the use of FPGAs still lags behind gate arrays and cell-based ASICs, according to R.H. Krambeck, C.T. Chen, and R.Y. Taui of AT&T Bell Laboratories, writing in a Compcon 93 paper. "For most of the 1980s, ... highly complex designs for which a large unit volume was expected were designed as cell-based ASICs," they said. "Somewhat simpler or lower volume circuits were done as gate arrays." The reason, they continued, was that FPGAs were inferior to the other two approaches in three key areas: gate density, system clock speed, and ease of use.

The FPGA is still a new product, less than 10 years old. Altera Corporation introduced one of the earliest versions in 1984. Containing eight macrocells, "the EP300 had a programmable I/O architecture as well as a programmable logic array," S. Kopec reported to Compcon 90. "This structure allowed a single device to implement arbitrary mixes of registered and combinatorial logic in a single chip."

By the time Kopec spoke, "the complexity and capability of programmable logic devices had increased by over 20 times." FPGAs would soon be competitive for many applications with cell-based ASICs and mask-programmable gate arrays.

Three years later Jesse Jenkins of Jenkins Research organized an all-day track at Compcon 93 to update progress in FPGAs. Papers from Altera Corp., Actel Corp., AT&T Bell Laboratories, Aptix Corp., Concurrent Logic Inc., Intel Corp., and VLSI Technology reported that gate-equivalent densities in the 5,000 to 10,000 range are now common. Several companies have exceeded 20,000. System clock rates range from 60 to 125 MHz. Clock-to-output delay is on the order of 10 ns. Pins run from 100 to 288.

With capabilities of this order, FPGAs can satisfy a large fraction of the fast-time-to-market applications. Craig Lytle estimates that Altera's new family, for instance, with densities up to 24,000 usable gates, is competitive in nearly 50 percent of gate-array design starts.

Some system operations have to be performed at a rate faster than a microprocessor, particularly a small, inexpensive one, can operate. For instance, "data capture is a hardware-intensive function," Jenkins observed. "It is difficult using software to synchronize controller operations with external data and balance memory requirements."

To match a microprocessor to a task of this sort Jenkins suggested:

- *Identifying tasks that must operate at full speed and those that can keep up at a slower speed.* For instance, an activity that must be completed in less than the cycle time of the processor has to be implemented in fast-response hardware.
- *Identifying operations that can be efficiently handled by microprocessor data paths and those that require high-speed logic in hardware.* An activity where speed is not critical is a candidate for handling by the micro-

0272-1732/93/0600-0073$03.00 © 1993 IEEE

processor with software.

- *Maximizing tasks that can be done by software and minimizing those that require hardware logic.*
- *Identifying asynchronous tasks that must operate with custom hardware because the pace required exceeds the interrupt capability of the processor.*

As an example, Jenkins outlined the design of a microcontroller accelerator for an instrumentation application. The design combines an inexpensive personal computer with an add-on hardware unit capable of capturing incoming data at a 40-MHz rate.

The potential Achilles heel in using large FPGAs for hardware add-ons, however, is ease of use. Converting logic equations, logic diagrams, netlists, schematics, hardware design languages, or other outputs of the logic-design process to the inputs needed to program the FPGA would be very time-consuming if it had to be done manually. To meet this need, all the companies have software tools, such as chip compilers, that provide largely automatic methods of performing this conversion. Some of these tools also interface with CAD tools supplied by outside vendors.

The output of these FPGA design tools is a data stream that programs the FPGA chip. Programming the chip means establishing the pattern of connections between the logic elements that configures the FPGA for its intended application. FPGA fabricators generally use two methods of making these connections: permanent and configurable.

Representative of the permanent connection is the antifuse element, employed by Actel, VLSI Technology, and others. This element is a dielectric material at the junction of two metallic lines. In its original state, it is an insulator and the connection is open. When programmed, by applying a high-voltage current to it, the material converts to a conducting state. This conversion is a one-time operation. It permanently programs the FPGA. If change becomes necessary, the designer would have to program a new FPGA.

Configurable FPGAs are programmed at power-up by downloading digital data to preset or clear flip-flops or other logic elements. In some applications the data stream is serial; in others, parallel. It may be obtained from a PROM or system RAM, or it may be downloaded from the system controller or microprocessor. One of Intel's FPGAs has a nonvolatile storage array on the FPGA chip itself that provides the data to configure the chip.

Configurable FPGAs are also reconfigurable by the simple process of loading a new set of configuration data. Reconfigurability has obvious advantages in adapting a chip to the needs of an application as they become apparent. It also makes it possible to operate a product in several modes, simply by reconfiguring its FPGAs between modes in a matter of milliseconds.

"With many different programmable logic solutions from which to choose, a design engineer is faced with an overwhelming task in effectively choosing the best programmable architecture to meet his needs," Jay Sturges of Intel said. The *Compcon 93 Digest of Papers* describes half a dozen of these possibilities.

Sturges, also in the same *Digest*, had a different objective. He outlined a quantitative approach to making a choice among the FPGAs available. He contends that there are many metrics, such as number of pins per module, ratio of fan-in to fan-out, and others, that help. Moreover, these metrics are mathematically related. For instance, when two are known, a third can be predicted. Consequently, a designer knowledgeable in these matters has a long leg up toward making a wise selection.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 186 Medium 187 High 188

Glitches left in software copyright system

Richard H. Stern

Oblon, Spivak, McClelland, Maier & Neustadt, P.C.

1755 Jefferson Davis Highway, Suite 400

Arlington, VA 22202

Maybe after the *Altai* and *Sega* cases you thought you no longer had any problems with copyright law interference when carrying on a career as a software professional. Think again. There are still plenty of features, glitches, or bugs (depending on how you see it) left in the software copyright system.

You will recall that in the *Sega* case the Ninth Circuit federal appeals court held that reverse engineering of software by disassembling code was permissible, if undertaken for "a legitimate purpose." In that case, the legitimate purpose was cracking the lock of a security system that kept "unauthorized" software out of the copyright owner's hardware platform. (The court did not explain what a legitimate purpose was for other cases. That will have to be determined by a possibly long and painful case-by-case process. The copyright statute does not address this issue, like most difficult points; therefore, judges must work out the answers in individual cases.)

In the *Altai* case, the Second Circuit held that copyright law had to be interpreted, to some extent, in the light of the needs of the computer software industry. That meant that software publishers' desires to protect "look and feel," "program structure," and "other nonliteral aspects" of software had to be scrutinized closely. This would occur when the effect of copyright protection would be to close off programmers' access to public domain features, functionally dictated aspects of programs, and features commanded by good software engineering practices. Also, programmers' "appropriation" of certain features from copyrighted computer programs would not be copyright infringement. So far, all to the good.

That stretch of software copyright law now seems more or less settled, for the time being. At least it is settled in the Ninth Circuit (West Coast states) for reverse engineering and in the Second Circuit (New York) for functional features. It will remain settled unless and until the losers in those cases talk the US Congress into legislatively overturning those decisions, which an IEEE document says they are trying to do. But there are other weak spots in the software copyright dike—and the heroic little Dutch boy, whoever that is in our parable, has only so many thumbs with which to plug holes in the dike. The propagation rate of holes may exceed the propagation rate of thumbs.

But the Ninth Circuit—praised for its *Sega* decision—has just created what may seem to be a new software copyright glitch. Loading software into RAM creates an infringing copy of the software and is therefore copyright infringement. That is what the Ninth Circuit just ruled (Apr. 7, 1993) in *MAI Systems Corp.* v. *Peak Computer Inc.* MAI manufactures and sells computers, creates and markets operating system software for the computers, and sells maintenance service for the computers. Several MAI employees left MAI to form Peak, a repair and maintenance service company. The usual trade secret, copyright infringement, and unfair competition suit followed—of which the copyright infringement aspect seems most troublesome.

MAI licenses its customers to use its system software. When Peak services a computer belonging to one of MAI's customers, Peak turns the computer on. That boots up the computer, loading the operating system software (from a hard disk, floppy, or ROM) into the RAM of the computer. Absent permission from MAI, the court held, that action makes a copy of the computer program, and it is copyright infringement.

Peak sought to defend on the ground that an

0272-1732/93/0600-0075$03.00 © 1993 IEEE

These are not glitches or bugs in the system. They are features!

infringing copy of a copyrighted work is made only when the alleged copy is "fixed" in a permanent or stable form, permitting it "to be perceived, reproduced, or otherwise communicated for a period of more than transitory duration." (This is a quote from section 101 of the Copyright Act.) Ephemeral copies, such as transitory images on a display screen, are not sufficiently "fixed" for them to be infringing copies under the copyright law. (The traditional copyright law analogy for this is that a poem written in the sand near the sea is not fixed in a copy, because it will be obliterated by the next tide.)

Peak argued, as probably most observers had believed thus far, that a copy in RAM is ephemeral, rather than fixed. It disappears when the computer is turned off, and the RAM is therefore no longer powered—the way the light from a light bulb disappears when you turn off the power. (At least, that is what happens in a PC. Perhaps, MAI's computers were left on all of the time—although that seems unlikely, unless they were mainframes.)

The Ninth Circuit disagreed, asserting that it is "generally accepted" that loading software into a computer creates a fixed copy of the software. The court was slightly troubled by the fact that the authorities it relied on for the alleged general acceptance did not reveal whether the software loading constituting the creation of an infringing copy was loading into RAM, a hard disk, or a ROM. No matter, the court said. A copy in a RAM can be "perceived, reproduced, or otherwise communicated." The court did not mention *for how long*, but it felt satisfied that "the loading of MAI's operating system software into RAM, which occurs when an MAI system is turned on, constitutes a copyright violation." Therefore, the court found Peak guilty of copyright infringement and enjoined its conduct.

To be sure, section 117 of the Copyright Act permits the owner of a copy of a computer program to load it into the owner's computer. But section 117 did not apply here, the court held, because MAI's customers are licensees of the software, not owners of copies.

In addition, Peak apparently owned several MAI computers and offered to loan them to customers. (Peak ran advertisements offering loaners to potential customers, for use while their own computers were being repaired.) The court held that this conduct would infringe MAI's exclusive right to distribute its copyrighted software, and the court therefore enjoined Peak from offering loaners—end of case, or this part of it.

Arguments can be made that this is a sound or unsound interpretation of the copyright law. For example, perhaps owners of computers have a right to get them repaired, by whomever they please—making the conduct of the repair company, when it acts at the behest of the computer owner, a "fair use" of the operating system software. Perhaps, computer owners have an implied license to do this, or perhaps MAI is estopped from preventing them from doing so, once it has sold them the computer. Perhaps, the asserted difference between selling and licensing the software in this context is a sham, and computer owners' property rights govern over legal fancy footwork. For example, if the ROM BIOS of the computer is licensed, rather than sold to the purchaser of the computer, should that legal label be taken seriously?

On the other hand, this may well be a proper interpretation of the US copyright law precedents developed over the last 200 years for poems, novels, paintings, music, sculptures, maps, and telephone directories—which govern here. These are not glitches or bugs in the system. They are features!

The point is that you can continue to expect interesting features of copyright law to turn up in software copyright cases, so long as copyright law is the statutory mechanism that we use to regulate software conduct. It would be unreasonable in the extreme to expect copyright law to be turned inside out, just to accommodate software professionals. Copyright law has a long and venerable history; it has its own traditions. To alter them just to suit software creators and users would run the risk of destroying traditional rights and benefits enjoyed by beneficiaries of traditional copyright law—publishers, artists, composers, motion picture studios, playwrights, poets, novelists, whatever. Indeed, to do that you might have to change copyright law so much that it would no longer be copyright law. (That is what the book publishers decided when the chip industry tried to get Congress to pass a chip layout copyright law. They kicked chip layouts completely out of the copyright law and put them under the *sui generis* Semiconductor Chip Protection Act of 1984, instead.)

Here's a different kind of glitch, although this one really deserves to be recognized as a deliberate feature of copyright law. The Eighth Circuit (North Central states) has just ruled (Apr. 6, 1993) in *National Car Rental System Inc.* v. *Computer Associates International Inc.*, that state law rather than federal copyright law governs the extent to which a software licensee may use a licensed computer program. CA licensed NCRS to use CA's computer programs "only for internal operations and for processing its own data." NCRS later started using the computer programs to process data of its truck leasing and car rental subsidiaries or affiliates. CA found out and threatened to sue.

NCRS then brought a declaratory judgment action, admitting the use, but

contending that such use was within the scope of the license and was not copyright infringement. CA then countersued, asserting breach of the license contract (a claim based on state law) and copyright infringement (a claim based on violation of federal law). NCRS riposted by asking the court to dismiss the breach of contract claim that CA brought under state law as being preempted by the federal copyright law involved in CA's second claim.

Section 301 of the federal copyright law provides that federal copyright law preempts state law claims when they are equivalent to copyright law claims. That means that state law cannot regulate the same thing that federal copyright law regulates. A state cannot authorize or prohibit copyright infringement, for example, because that would interfere with copyright law's uniform regulation of the field.

When is a state law equivalent to copyright law? That is the central problem in copyright preemption cases. Typically, the case turns on whether the state law claim has one or more extra elements in it (such as use of force, carrying away of property) that are not necessarily found in a federal copyright infringement claim, and are qualitatively different from the elements of the federal copyright claim.

Here, the trial court held that CA's copyright infringement claim was the same as its breach of contract claim, and it therefore dismissed the latter. The trial court said that both of CA's claims were directed to the same thing, unauthorized utilization of the copyrighted computer program, in this case amounting to a computer program lease going from NCRS to its affiliates.

The Eighth Circuit found the controversy to raise a close point on equivalency. But it concluded that the provision in the license contract between CA and NCRS limiting how NCRS was to use the computer program constituted an extra element not found in a copyright infringement claim. The Court thought this would be true even though a use of intellectual property in excess of the scope of a license is infringement. Here, the contract goes beyond CA's rights under the copyright laws. It requires NCRS not to use the software for the benefit of third parties. Accordingly, the Eighth Circuit reversed, holding that the contract claim under state law should be tried along with the federal copyright infringement claim.

Probably, the Eighth Circuit read copyright law properly. Congress did not intend to make copyright law cut off contract claims, in the ordinary case. Sometimes, contract claims and copyright infringement claims are equivalent. But these claims are probably not equivalent, because of a qualitatively different extra element.

What is the result, however? Consider a software company that has a nationwide marketing policy and a standard contract that it uses throughout the United States. Whether restrictions in it, such as the use restriction involved in this case, are valid and enforceable depends on the laws of 50 different states. Perhaps that is a rational choice for the ordinary subject matter of copyright law. But it is not a sensible way to regulate a nationwide software business.

It would make more sense if customers' violations of restrictions in software licenses were judged under a federal standard. They should be held copyright infringement if the vendors' restrictions are legitimate from a standpoint of software policy. They should be held permissible conduct (not violations of state or federal law) if the vendors' restrictions go too far from the standpoint of a federal law's software policy.

That is not the present law, of course. That could be the law only if Congress passed a uniform software regulatory law for the whole country. One provision of it would be a rule on when restrictions on customers' use of computer programs were permissible and thus customers' disobedience of the restrictions were infringement. That would provide a uniform federal system regulating software conduct. It would provide known, predictable remedies for violation of the regulatory standard with resulting certainty and security of expectations for software users and software marketers.

We need a system whose features don't look like bugs.

There are many pros and cons of having such a regime, and the arguments generate considerable heat. (A professorial representative of one important mainframe company has just published a lengthy article heatedly denouncing *Sega, Altai,* and the "false icon of *sui generis* protection" for software. He argues that it would be a great mistake to adopt a "genre-specific *sui generis* regime invented out of whole cloth" as an alternative to the "growing experience under the present copyright law and the increasing certainty that it provides" for software as "court decisions by degree crystallize into an understandable and sensible doctrinal matrix.")

From time to time in the coming months, I will address various aspects of such issues and comment on different observers' positions. Much later this year, I will report on a conference at a major national school of engineering. This conference will be devoted to the issue of whether it is time to explore alternative models of software protection law. Such altemative models would differ from the pure patent law or copyright law models (as I prefer to term them) or paradigms (as many of my nonengineer professor friends prefer to term them). Perhaps the time has come to consider reinventing the form of intellectual property or industrial property protection that we use for software, or for some kinds of software.

Perhaps that could lead to a system whose features did not look like bugs.

Guest review

Of general interest to readers is the following book review by John F. Noble, Washington, D.C., attorney and editor in chief of the *Computer Law Reporter*, a monthly journal of computer law and practices. Noble's full comments on the current battle for control of the software industry appeared in the April 1993 issue of *CLR*; what follows is a shortened version.

Softwars: The Legal Battle for Control of the Global Software Industry, Anthony L. Clapes (Quorum Books, Westport, Conn., 325 pp.)

A war rages over the extent to which the original developers of computer software should be protected from the sincerest form of flattery. Both sides of the battle will enjoy Clapes' account of this high-stakes war. Unencumbered by a lot of technical and legal jargon, Clapes mixes a litigator's war stories with a cogent philosophy of what the law should be. He adds just enough economic analysis to anchor the debate in the real world.

Clapes is not an unbiased correspondent. As IBM's assistant general counsel in charge of litigation, he is one of the warriors, and his allegiances are obviously and admittedly with the "innovators" in their campaign against the "imitators." This does not detract from the book. Indeed, an author pretending to even-handedness could not have portrayed the fierce pitch of the battle that is conveyed by Clapes' sometimes acerbic account. At the risk of pushing the metaphor, you can almost smell the blood.

Softwars recounts the development of the law regarding the application of patent, trade secret, and particularly copyright protection to computer software, from *Apple* v. *Franklin* in 1982 to *Nintendo* v. *Atari* in 1992. In the course of this exposition, the author takes the reader to courtroom battlefields in Philadelphia and Northern California, across the ocean to Japan and Australia, and into cyberspace where a would-be astronomer tracks down a pirate named Hunter in the employ of the KGB.

The theme of the book is that "the outcome of the legal battles over software will determine the nature of the industry for the foreseeable future, and the nature of the industry will dictate the identity of the firms that will be most successful in competing in that industry." As its author says, "The prize is the computer industry itself."

This war, according to Clapes, will determine whether the computer industry in the future is marked by innovative competition between firms that rely on the development of advanced products, or imitative competition between firms that compete primarily on price.

Clapes weaves legal and economic arguments in support of the proposition that the vitality of the industry depends upon providing legal protection to the creative work of programmers and the capital investment of their sponsors.

The legal argument proceeds from the premise that the creative nature of the work, and Congress' explicit extension of copyright protection to computer programs, entitles software to protection under the traditional principles of copyright law. Under those principles protection extends to the nonliteral elements of original expression, and applies in particular to the program's "interface" with the user.

The seminal case for this "traditional" application of copyright law principles to software was the 1987 decision in *Whelan Associates, Inc.* v. *Jaslow Dental Laboratories, Inc. Whelan* held that an accounting program for dental laboratories was infringed when a competing program adopted its "sequence, structure and organization." In that case, the Third Circuit Court of Appeals applied the principle that only expression and not the underlying idea of a work is protected by copyright. It held that the unprotectable idea of a utilitarian work like a program is its purpose or function, and that the protectable expression was everything that was not necessary to the purpose or function.

Clapes acknowledges that the case was "highly controversial." But he insists that the decision represented "nothing more than the application of traditional copyright principles to a case involving computer programs." He is not the least embarrassed to defend the decision in *Whelan*. Still, one might wonder about his characterization of the decision as a "victory, *some conclude* for the purveyors of proprietary software." Would he agree that it has turned out to be something of a Trojan horse? In the subsequent case law and commentary, its analysis of the idea/expression dichotomy has almost unrelievedly been described as simplistic and overbroad.

For a time, it seems, the "innovators" were on the offensive. Clapes applauds Judge Keeton's decision in *Lotus* v. *Paperback*, decided in 1990. Keeton held that the copyright law does not treat computer programs differently. Therefore, nonliteral elements of the 1-2-3 spreadsheet program, such as its menu structure, were entitled to protection under traditional copyright principles.

Clapes recounts the reaction to Keeton's "lodestar" decision as the opposition regrouped. The so-called "gang of ten" copyright law professors rose up to advise the Court that its copyright analysis was contrary to the statute, case law, and traditional principles of copyright. The League for Programming Freedom picketed Lotus' headquarters. Law firms with clientele in the enemy camp "salted" the legal literature with articles critical of the opinion in *Lotus*. One commentator, according to Clapes, claimed that the judge had given Lotus the exclusive right to "use the F1 key for a Help function."

Forward in time, and across the country, Clapes turns his attention to *Apple* v.

Microsoft. In that case, Judge Walker rejected the claim that Microsoft's Windows appropriated the overall look and feel of the Macintosh user interface, instead examining isolated elements of the Mac interface for protectability.

Some might conclude that the evil empire of imitators was resurgent. But Clapes offers a more sanguine analysis. He points to an earlier license agreement between Apple and Microsoft. Clapes argues that "just as a separation agreement complicates a couple's lives, the license agreement complicated Apple's prosecution of its case."

As a result, according to Clapes, the judge concluded that he was required to separate the licensed elements of the interface before comparing the two works. According to Clapes, "[N]ormally, a copyright plaintiff, particularly one who is asserting copyright in graphical images, is entitled to have an infringing work evaluated against the original work on the basis of the *gestalt* of the two works: the overall impression or 'total concept and feel' that they convey."

There is no particular citation to support this broad statement of the law, and the issue is a closer one than Clapes will allow. One can almost hear the "gang of ten" collectively muttering, "in your dreams, pal."

Clapes would reserve a special place in hell for the advocates of "reverse engineering." This defense to appropriation of intellectual property found favor with the Supreme Court in *Bonito Boats, Inc.* v. *Thundercraft Boats, Inc.* The *Bonito Boats* decision stated that the reproduction of a fishing boat hull design, claimed as a trade secret, by making a mold of the original was not unlawful. It was legal because "[t]he public at large remains free to discover and exploit the trade secret through reverse engineering of products in the public domain."

The reverse-engineering defense has been invoked in the software context as a defense to the practice of discovering the programmer's trade secrets by running a program's object code through a reverse compiler to replicate, or at least approximate, the original source code. This, complains Clapes, allows a competitor to "unlock the secrets of an original program" by "peeking."

Clapes argues strenuously that reverse compiling is different from the kind of reverse engineering approved by the Court in *Bonito Boats*. The shape of a hull enters the public domain when it leaves the factory. Unlike this, a computer program, on the other hand, keeps its secrets by being marketed in machine-readable object code, and sold subject to license agreements that prohibit reverse compilation. According to Clapes, "'Reverse engineering' is a term that makes no real sense when applied to software. The use of the term is a form of propaganda that obscures what is really going on."

In Clapes' view, reverse compiling a program is indistinguishable from translating a French novel to English—a right reserved to the author.

He undermines his own argument, however, by attempting to rebut the antiprotectionist argument that reverse compilation is necessary and appropriate to the understanding of the program. Understanding is necessary to the dissemination of its ideas, which is in turn one of the bedrock purposes of copyright law. Although he takes issue with this articulation of the purpose of copyright law, his fallback position is that object code is not unreadable at all.

If Clapes is right that "[s]ome programmers can read object code[,]" and that [m]any more can decipher it with effort," it would seem that he is in the same boat with *Bonito*. His secret is not so secret. The legitimacy of the endeavor to uncover the "secret" surely does not turn on the arduousness of the task.

Clapes tours battlefields in Australia and Japan where the reverse engineers were routed by AutoCAD and Microsoft. His description of the Japanese legal system is informed and fascinating. His account of the Australian case about a hacker who bypassed AutoCAD's security system proves that Clapes is not totally shameless in his protectionist sensibilities. The hacker was found guilty of infringing a copyright in an electrical circuit.

"The prize," says the author, is the computer industry itself."

The Gettysburg of the Softwars is *Computer Associates* v. *Altai*, a case that has assumed a significance far surpassing the initial stakes.

The case concerned an interface program called Adapter, developed by Computer Associates and copied by an Altai employee.

The case would have been unremarkable but for Altai's response to CA's copyright infringement and trade secret misappropriation complaint. Altai rewrote the program to eliminate the portions copied verbatim from CA. Oscar 3.5 was the result.

CA amended its complaint to allege that Oscar 3.5 was also an infringement of its copyright and a misappropriation of its trade secrets. Because there was no direct evidence of copying, the Court was required to determine whether Oscar 3.5 remained "substantially similar" to Adapter.

Judge George C. Pratt, a member of the Second Circuit Court of Appeals on temporary assignment in district court was the trial judge. He began what Clapes refers to as his "revisionist analysis" of the substantial similarity issue by observing that "in the context of computer programs, many of the familiar tests for similarity prove to be inadequate, for they were developed historically in the context of artistic and literary, rather than utilitarian works."

Clapes' understated characterization

of this premise as "questionable" does not totally convey the dismay it must have engendered in the protectionist camp. And it got worse. As recounted by Clapes, "[f]rom his shaky premise, Judge Pratt leapt to the even more tenuous conclusion that the *Whelan* case, without question one of the most important software copyright cases decided to date, was 'inadequate,' 'inaccurate,' 'simplistic,' and 'fundamentally flawed.'"

Leaving *Whelan* like an overturned tank by the side of the road, Pratt formulated what has become known as the abstractions-filtration-comparison test. That test is based on Judge Learned Hand's 1930 opinion for the Second Circuit in *Nichols* v. *Universal Pictures Corp.* It recognizes in a work "patterns of increasing generality" as "more and more of the incident is left out," leading up to "the most general statement of what the [work] is about." Under this test, the court is called upon to identify a level of abstraction that will be the demarcation between protected expression and unprotected idea.

When Pratt applied his abstractions test to the Adapter program, what remained of Adapter in Oscar 3.5. Its parameter lists, macros, and "high-level architecture" fell, for the most part, on the unprotectable "idea" side of the dichotomy.

Clapes labors to minimize the significance of the decision in *Computer Associates*. He claims to be heartened by the fact that the Court, by citing *Nichols*, reached back to the "wellspring of presoftware precedents dealing with the idea/expression dichotomy." Clapes questions whether there are really "essential differences" between the two tests.

It may seem that Clapes is whistling past the graveyard here. After all, under the first test "structure, sequence and organization" is protected. Under the second, parameters, macros, and high-level architecture are not.

But I suspect that Clapes is more devious than oblivious. He is arguing his case, in a well-crafted brief, before the judges and their clerks in nine other circuits, and one higher venue, which have yet to squarely address the issue.

The choice that Clapes would *like* to present is between *Whelan* and *Nichols*, not *Whelan* and *Computer Associates*. He is right that *Nichols* and *Whelan* are not that far apart, and the protectionists would be delighted to have the battlelines drawn there. The problem with this is that Pratt's decision merely uses *Nichols* as a legitimate point of departure, and he is a far piece down the road before he is finished. The antiprotectionists have advanced, and a new battleline is drawn between *Whelan* and *Computer Associates*.

As much as he might wish it were not so, Clapes is aware of this too. He takes some shots at Pratt's opinion, and the factors that produced it, in a effort to undermine its authority.

In particular, Clapes suggests that the outcome was unduly influenced by the technical expert—Randall Davis of MIT's Artificial Intelligence Laboratory—appointed by Pratt to help him interpret the technical evidence. Clapes credits the expert as being an "intelligent, articulate, and interested computer scientist ... known as a moderate" in the software protection debate. However, Clapes maintains that "everyone has a bias, and Davis is no exception." Davis' bias, in Clapes estimation, is that of a skeptic, inclined, in Davis' own words, to the view that the "old ways of doing business and the old ways of thinking may simply not work any more."

Clapes argues that Davis, in his unusual but not unprecedented role as an independent expert appointed by the Court, swayed the outcome of the case. He suggests that Davis' opinions were given greater weight because of his ostensible neutrality. Also, his bias toward "questioning the fundamental premises of intellectual property law" was not fully revealed at trial because the attorneys felt constrained to "treat him fairly gingerly."

Clapes also suggests that Pratt's "revisionist" analysis was born of an inclination to seize the opportunity to assert the Second Circuit's reputation as the strongest copyright law circuit. He wonders: "Was the whole point of the rejection of the *Whelan* analysis to promulgate a Second Circuit test for nonliteral infringement instead of a Third Circuit test, a kind of Not-Invented-Here reaction? Perhaps." Clapes points out that Pratt's brethren on the Second Circuit, "not surprisingly," affirmed his decision.

Unfortunately, Clapes' account of what may prove to be the 100 Years Softwar apparently bumped up against his publisher's deadline. The Federal appellate decisions in *Atari* v. *Nintendo* and the Ninth Circuit in *Sega* v. *Accolade* receive only cursory treatment in a footnote.

If one makes appropriate allowance for the partisanship of its author, *Softwars* is a valuable survey of the battlefield's terrain. Newcomers to the field of intellectual property law will find it accessible. For lawyers already steeped in intellectual property law, the book provides a thoughtful and provocative perspective.

If the book has a failing, it is that Clapes' orientation does sometimes color his depiction of reality. At one point, for example, he writes: "There is no real uncertainty about copyright protection for software. Don't let anyone tell you differently." It is true, of course, that software is entitled to copyright protection. But this would not have been nearly so interesting a book if there were not a great deal of uncertainty about the *extent* of copyright protection that software is entitled to, and the *nature* of the conduct that will be deemed infringing.

Micro Standards

Organizing the corporate standards function

Stephen L. Diamond

SunSoft, Inc.

Phone (415) 336-4190

Fax (415) 336-4477

steve.diamond@eng.sun.com

In today's world of "rightsizing" (a euphemism for downsizing), "downsizing" (a euphemism for layoffs), layoffs (a euphemism for not getting a salary), "leftsizing" (watch this space for details), and every other kind of "sizing" but "upsizing," standards participants are increasingly being called upon to justify their existence with short-term economic benefits. At some particularly shortsighted companies engaging in this sport du jour, a strategic rationale for standards without a "this-quarter-bottom-line-dollars" benefit is just short of useless.

While this situation is undoubtedly painful for some standards participants, I believe that it represents an opportunity to refocus a company's standardization efforts on their real function—helping to make the company's products successful. Such a focus on product success emphatically does not mean attempting to subvert the standards process for parochial ends. On the contrary, it means a commitment to working with standards processes over the long term, as an integral part of the development of successful products.

An analogy that I find useful is product quality. Attempting to achieve quality by bolting it onto an existing product is absurd—quality must be an integral part of the entire process of the company. The same is true of standards: decisions regarding participation in the development of standards, conformance to standards, or the creation of standards cannot successfully be treated as afterthoughts. To do so encourages terrible standards and unsuccessful products. Standards, like quality and the spices in spaghetti sauce, must be in there from the beginning.

There are many ways to organize standards within a larger company; some are very productive and others are not. It's very difficult, for example, to internalize the "in there" approach where a centralized standards organization calls all the shots on standardization issues. In such a situation, the product groups—the engineering and marketing departments—either abrogate their product responsibilities and let the standards department dictate standards compliance and participation, or they ignore the standards organization entirely. Both outcomes are potentially disastrous while being wholly avoidable.

The key is matching the standards organization to the developmental stage of the corporation. Paradoxically, smaller companies often do a better job in standards than larger companies, at least in the short term. Because they usually can't afford to hire standards specialists, the engineering and marketing departments must be involved directly. Unfortunately, these companies often do not have the resources to make the necessary long-term standards investments.

Properly managing standards involvement in larger companies represents a "win-win" opportunity for both the company and the industry within which it operates. In this column I propose a three-stage internal standards organization developmental taxonomy (see Figure 1).

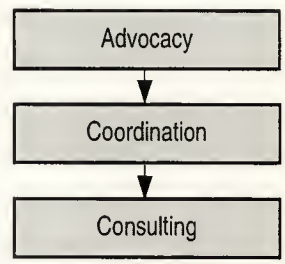


Figure 1. Internal standards organizational taxonomy.

0272-1732/93/0600-0081$03.00 © 1993 IEEE

Standards advocacy

Larger companies may reach a significant level of standards involvement before they consider creating a standards department. Sometimes hundreds of engineers are participating in standards before a company recognizes the need for some focus. The good news is that people in the product groups are involved with standards because their products depend on them. On the other hand, such involvement in standards lacks any long-term commitment or focus and has little or no coordination. Also, little thought is usually paid to effectiveness or efficiency.

At this stage, the standards function should begin with a small team focused on advocacy. By "advocacy" I don't mean preaching standards as a panacea, rather the team should educate the product groups about the universe of standards development activities. This education should be quite focused, and cover standardization status, process, trends, and potential rewards. Product development groups should be aware of which standards organizations and activities impact their current and future products. They should understand how standards are developed, and how they can become involved. Finally, they should have a picture of future trends so that they are not blindsided when a standard is being developed or approved in their sphere of interest.

The standards team in the first stage might consist of four people: an engineer to serve as a liaison to the engineering community; a marketer to serve as a liaison to the marketing community; a standards expert; and a manager to bring the standardization message to company management. To avoid expectations that the standards team is there to "do standards," it is important not to build a large group. The product groups should "do standards;" the standards group is there to navigate—not to drive. If it becomes necessary to participate in a standards developing organization (SDO) creating a standard that the company wishes to implement, the resources should come from the product groups, not a standards group. The goal of stage one is for the product development groups to achieve the "minimum adult requirement" of understanding of the standardization process, and to begin to address standardization requirements in all their product definitions.

"Address standardization requirements" doesn't mean that every product must conform to a standard. It means only that every product definition has to consider standardization, both formal and informal. It may mean that the product absolutely does not conform to an existing standard, but it may also mean that the product team will have to participate in the development of a future standard. To address standards, the team must understand them. Team members must be able to assess the costs and benefits of the standardization equivalent to the make/buy decision: the lead/follow/ignore decision.

Every product definition has to consider standardization, both formal and informal.

Where a standard exists, the product team has to choose whether or not to conform. Even if they choose to conform, team members still must decide whether to participate in the evolution of the standard in question, or simply to sit back and observe. Where no standard exists, the product team has to decide whether or not to lead in the creation of a new standard. Another possibility is to ignore formal standards and attempt to create an informal, de facto standard, but the issues to be addressed are very similar. My point is that the product team must explicitly consider these decisions.

Standards coordination

Once the suggested minimalist standards infrastructure is in place, standardization activities ideally will begin to diffuse throughout the company. At this stage, the standards function should move from an advocacy role to a coordination role, with both an internal and external perspective. Internally, the now-standards-literate product development people will increasingly begin to make decisions regarding standardization. These decisions should make sense across the company; externally, with the plethora of standards developing organizations today, it is essential that a company's standards efforts are coordinated across organizations. For example, consortia, user groups, and SDOs overlap significantly in scope; one organization may attempt to develop a standard that overlaps with one developed elsewhere.

Standards consulting

Significant, distributed standards awareness and competence within the product development groups characterize the third stage. The standards function should metamorphose at this point into a proactive high-level consulting organization. Having liaisons into the engineering and marketing communities may no longer be necessary, because these communities will have already embraced the message and their need will be for specific information and consulting. Instead, standards specialists should replace these liaisons in the standards department. Maintaining a high level of management awareness of standards, however, is still important, preferably by a high-level management reporting relation-

ship. The ultimate function of a standards department should be to

- educate management, engineering, and marketing;
- provide information beyond the company's usual sphere;
- monitor and project standards trends;
- guide internal standards participants;
- consult; and
- help coordinate broad, long-range standardization strategies.

Standards people typically should not hold technical positions in standards bodies, I suggest, but rather should arrange that appropriate representatives from a product group be made responsible for standards by their management. On the other hand, a key function of the standards professionals at this stage is to participate in high-level policy-making positions within standards bodies. There are several reasons for this commitment. First, I believe that beneficiaries of the external standardization infrastructure have an obligation to work at improving it for the benefit of the industry. Second, such positions provide greater visibility and access and indirectly help the company. Finally, by virtue of their role within the company, the standards professionals are in the best position to encourage a fair and open process.

The meaning of time

A key factor that must be considered in the development of an internal standards organization, processes, and strategies, is time—not in the tactical sense but in the historical sense. Most standards organizations, especially SDOs, have evolved over years. The IEEE, for example, has been involved in standards development for over 100 years. Any company that expects to participate meaningfully in standards should treat that involvement as a long-term commitment, both to standards and to the industry. Companies that are willing to make such a long-term investment will help themselves and their industry. Companies that seek to achieve short-term, parochial advantage in a long-term standards world will benefit neither.

Obvious benefits

In conclusion, let me raise the question of why standards participants are so often asked to justify their existence. Aren't the benefits of standards obvious even to management? The short answer is no. The long answer is that standards professionals, thinking that the benefits are obvious, often fail to adequately communicate them to their company's management. Another problem is that standards professionals frequently are reluctant to involve themselves in what they perceive as petty "commercial" issues, preferring to deal in the rarefied realm of standards organizations and their often hermetic and arcane policy, procedures, processes, and people. The solution is simple: by bringing standards into the product development process as an important and integral participant, the benefits of standards become obvious to everyone.

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 180 Medium 181 High 182

Manuals and guest reviews

Richard Mateosian

2919 Forest Avenue

Berkeley, CA

94705-1310

(510) 540-7745

Nowadays it is entirely feasible for a small company or product development group to produce its own highly professional hardware or software manuals. One tool stands out as the best for this kind of job.

FrameMaker 3.0 (Frame Technology Corp., San Jose, Calif.)

I've used Microsoft Word for the Macintosh for many years, and recently I've become very fond of Word for Windows on PC systems (See Micro Review, Feb. 1993). For many jobs I wouldn't use anything else. For large jobs, however, the nod has to go to FrameMaker.

Unlike Word, which started on small systems and added features over time as those small systems became more powerful, FrameMaker began as a publishing system on workstations. As personal computers have evolved into sufficiently powerful systems, FrameMaker has migrated unchanged and intact to them. While preparing for this review, I worked with both Macintosh and Windows versions; they are virtually identical. Files are interchangeable between them. Since large jobs often require the collaboration of several writers using a variety of computers, this is an important feature. By contrast, Word for the Macintosh and Word for Windows have separate histories, but have evolved into similar programs. They are far from identical.

Another interesting difference between Word and FrameMaker arose when I tried to use each to read the other's files. FrameMaker easily opened a Word document and displayed it with all its fonts and formatting. Word, on the other hand, opened a FrameMaker document, and it looked like pages of gibberish.

In my review of Word 5.1 for the Macintosh I complained about how difficult it was to use on the tiny screen of a Mac SE/30. This problem is far worse for FrameMaker. You can use it on a small screen, but it's practically impossible to work effectively there. In fact, it's difficult to use on the larger 640×480-pixel VGA display of my PC. The other side of this coin is that FrameMaker can make good use of, and justify the purchase of, the high-end machines of the PC and Macintosh lines.

FrameMaker allows you to make a book file that defines a book as a sequence of separate files. You can design page layout and paragraph and text formats for the book then use FrameMaker's capabilities to enforce uniform formats and continuous numbering of pages, figures, tables, and other sequences. You can build cross-references, an index, and a table of contents, and FrameMaker will update these for you whenever you ask it to. These tasks are easy to accomplish with FrameMaker, once you master its arcane ways of specifying them. They are almost impossible to accomplish with Word.

FrameMaker has a built-in drawing facility, and it can import graphics in a large variety of formats, either by actual inclusion or by reference. Importation by reference uses paths relative to the directory that contains the book file, so it is easy to keep the files together and move the entire package from one system to another.

Since FrameMaker maintains a uniform environment across platforms, its Macintosh and Windows versions don't look like standard Macintosh or Windows programs. This can be disconcerting. On the other hand, there are platform differences in areas like keyboard shortcuts. These can make switching between platforms a little rougher, but it's still pretty smooth.

The biggest lack I find in FrameMaker is a good macro facility. By comparison with

0272-1732/93/0600-0084$03.00 © 1993 IEEE

Microsoft's Word Basic, FrameMaker's capabilities are virtually nonexistent.

The other complaint I have about FrameMaker is that I often can't predict what it will do in response to what I think are reasonable commands. The worst examples I've seen of this involve trying to place and group graphics in anchored frames. Pieces of them sometimes wind up scattered all over the printed document. Either I don't understand how it works, or it has a few bugs. The whole program is so complicated that I'm not sure which is true. The documentation, while voluminous, has many gaps when it comes to details.

If you want to work across platforms or put together large documents, FrameMaker is the first product you should consider. It may well be worth the extra money it will cost you.

Compression

In the interest of disclosing any possible conflict of interest, I need to tell you that AddStor, Inc., a major supplier of compression products for PCs, is one of my technical writing clients.

Compression is the next big thing. It's an idea whose time has come. There is never enough disk storage and never enough transmission bandwidth. Every disk soon becomes too small, as programs and data expand to fill the available space. Every modem and every network soon becomes too slow, as larger programs and more data move from one place to another.

The first attack on this problem came from the proliferation of archiving programs like Stuffit and PKZIP. These simply take a collection of files and build a large file containing all of them. By encoding redundancy and eliminating the dead space at the ends of the individual files, the archiving program builds a combined file that takes up a lot less room than the sum of the individual file sizes.

As the popular personal computers became more powerful, they made possible another kind of compression. This is exemplified by PC programs like Stacker and SuperStor, which perform compression and decompression in real time. These programs compress virtually all files on a disk into a single large file and install a driver program to allow accesses to that file as if it were a disk drive. The driver compresses as it stores and decompresses as it reads, and it all happens so fast on processors like a 486 that you never really notice the delay.

The main problem with programs like those described in the last paragraph is that you need to add them onto the operating system through the CONFIG.SYS and AUTOEXEC.BAT files on the PC or through similar mechanisms on other systems. The obvious next step was to include compression as an integral part of the operating system, and Microsoft has just done that.

MS-DOS 6 (Microsoft Corporation, Redmond, Wash., $129.95)

MS-DOS 6 is the next major version beyond MS-DOS 5. Its main achievement is to incorporate features that large numbers of DOS users were adding to their systems from third-party sources. It includes virus protection and backup utilities that run from Windows, a memory manager, several levels of file-deletion reversibility, and compression.

The compression system included with DOS 6 is called DoubleSpace. You simply invoke the installation program from the DOS prompt, wait a few minutes for it to compress your files, and with no apparent change to anything else, you now have about twice as much disk space as you had before. I did this about a month ago, and I have had absolutely no trouble running all of my old DOS and Windows applications.

Microsoft claims that DOS 6 is more tightly integrated with Windows than DOS 5 was. I haven't noticed any difference along those lines. I certainly haven't had any trouble running Windows.

I can't think of any reason why you shouldn't run right out and upgrade to DOS 6.

Books

High-Speed Digital Design—A Handbook of Black Magic, Howard W. Johnson and Martin Graham (Prentice Hall, Englewood Cliffs, N.J., 1993, 458 pp.; $45.95)

When I received this book a couple of weeks ago, I didn't look at it very carefully or really notice it. Then I ran into Marty Graham at a conference and saw him waving it around proudly, and I finally realized what I had received. After 40 years in this business, this is the first book U.C. Berkeley professor emeritus Graham has ever put his name on. If you are interested in high-speed digital design, you should drop everything and read it.

I don't know how the authors divided the creation and organization of the topics. Graham told me that Johnson, a specialist in high-speed digital communications and digital signal processing, did all of the writing. He also told me that every example came from the direct experience of one or the other of them. In other words, this is not a survey. It documents the experience of two respected specialists in a neglected field.

This book aims to alleviate a problem in the education of digital designers. Over the last couple of decades, the analog circuit principles that apply to high-speed digital design have fallen out of standard college curricula—for the simple reason that they are largely irrelevant at the speeds most designers have been working with. Now, however, as speeds increase, designers lack the training to deal with the "black magic" of managing high-speed effects.

You can learn the basic terminology, the high-speed properties of logic gates, and the standard measurement techniques by reading the first 130 pages. After that you can skip around among the specialized topics that make up the rest of the book.

I'm not an expert in this field, and I haven't read the whole book, but I sampled every chapter. The clarity of the text and the excellence of the dia-

grams impressed me. I also liked the editing and the formatting, with one exception. The publisher chose to indent every paragraph, except for the first paragraph in any first-level or second-level section. This makes passages that contain short paragraphs, centered formulas, and small diagrams difficult to scan.

I especially liked one feature of the book. Every few pages the authors have included boxes containing "points to remember." They have collected these into a nine-page checklist for system design at the end of the book. The checklist includes the chapter and section of each point, so it functions as a high-level outline of the main topics of the book.

If you aren't doing high-speed digital design now, chances are that advances in technology will soon move you into that bracket. Read this book now and avoid grief later.

Intel's SL Architecture—Designing for Portable Applications, Desmond Yuen (McGraw-Hill, N.Y., 1993, 345 pp. plus diskette; $39.95)

In one of my early columns (April 1987) I looked at the then current state of manuals. We've come a long way since then. In those days a microprocessor supplier could only dream of putting a book of this quality into the hands of designers. Now they not only do so, but designers willingly pay $40 for it. The effective use of outside publishers has been a large factor in making this happen. The book combines the inside knowledge of a senior Intel applications engineer with the publication expertise of a major publisher.

Yuen's book deals with how to use Intel's 386SL and 486SL CPUs and the companion 82360SL peripheral controller. Intel designed these chips to facilitate the design of portable computers. Their key feature, system management mode (SMM), addresses the central issue in this field, battery life.

Yuen takes you methodically through all aspects of designing with these chips. It's like a complete collection of application notes with a coherent organization and a stronger than usual emphasis on principles. The associated diskette contains all of the sample programs. It even contains a debugging program and all of the register configuration files you need to use it.

If you are going to design with these chips, you need this book.

Guest book reviews

Our sister publication, *IEEE Software*, regularly solicits book reviews from a large pool of computing professionals. Recently, their backlog has grown. Many of these reviews are of books that *IEEE Micro* readers will find interesting. To help these reviews reach the public in a timely manner, we've included two of them here. If you have friends who read *Software* and don't usually receive *Micro*, I hope you'll show them a copy. Be sure to point out to them that subscriptions to *Micro* remain a great bargain.

Software Engineering: A Programming Approach, 2nd ed., Doug Bell, Ian Morrey, and John Pugh (Prentice Hall, Hertfordshire, England, 1992, 338 pp.)

Reviewer: J.E. Jordan, National Research Council, Ottawa, Canada

This easy-to-understand, nonmathematical overview of software engineering targets undergraduate students and software practitioners who wish to keep abreast of developments in the field. The book is a good example of the adage, "Small is beautiful." Concisely written in just over 300 pages, it is available in inexpensive paperback format but provides an extremely well-written, enjoyable, and readable treatment of the subject. Indeed, the authors succeed in conveying more usable information than many other more comprehensive, multivolume treatises.

The chapter on formal methods is a good example of a comprehensible treatment of a difficult subject. All of this should be welcome news to those who want to learn important concepts in the field but who have limited time and money.

The material is aimed at satisfying the requirements of CS14: Software Development and Design in the ACM curriculum. It involves traditional techniques of software development with an emphasis on programming language and graphical methods.

This second edition includes material on new techniques such as object-oriented programming, parallel programming, formal verification, and issues of programming "in the large." The book's main strength is that it provides a meaningful look at a number of design methods and programming paradigms. The section on design is particularly worthwhile, illustrating functional decomposition, data decomposition, and object-oriented approaches. The comments on programming paradigms, also well-written, detail the different languages and philosophies available to developers.

Though quite interesting and relevant to software engineering, the final section on implementation is more of a grab bag, including software tools, validation and verification, fault tolerance, and programming teams.

While software development management is not covered extensively, the last chapter on programming teams does discuss one aspect of this topic. The book emphasizes technical issues such as programming languages and programming environments, which are probably appropriate for an introductory text for use in a computing science curriculum.

I recommend this book as an excellent introduction to software engineering and comparative programming language studies suitable for undergraduate students or practicing programmers who wish to keep current with recent developments.

Statistical Methods for Testing, Development, and Manufacturing, Forrest W. Breyfogle III (John Wiley

and Sons, N.Y., 1992, 516 pp.; $64.95)
Reviewer: John W. Horch, The Horch Company, Madison, Ala.

This is the statistical reference book I've been waiting for. It is full of examples, contains simple text (to the extent possible for this topic), and includes a guide to use of the material. The statistics are mathematically correct, as I would expect. But the real value of the book is in the discussions of appropriate application of the statistics and the recognition of recent work in process improvement (Taguchi, Motorola, and others).

Before the actual text begins, Breyfogle guides readers toward their special interests. For example, the first table defines 30 or so areas of interest and the chapters dealing with them. As a software practitioner, I followed the directions and had great success when I was led to seven chapters that deal with the kinds of statistical methods applicable to software activities.

Breyfogle's second goal was "to make the topics practical to such an extent that this reference guide would become worn out." I believe this will be the case. Especially now as statistical methods are being bantered about in the software world, this book will be tremendously valuable to those who are charged with applying statistics to their work.

Achievement of the third goal, "to sell employees and all levels of management on the power of wisely applied statistical concepts," remains to be demonstrated. Readers may get a better understanding of their application with this book; however, wise application may be less achievable. My experience is that each "new" technique is adopted without much thought to the reason for the adoption. We need to be selective about the applications rather than blindly going forth, having limited success, and losing management's commitment because we lack useful results.

After this obligatory discussion, Breyfogle gets right to the point: "define the problem or question you want to answer." For a requirements bigot like myself, this is good news.

The heart of the book is the second section. The author reiterates the need to define the problem and then describes the analysis being applied. A later "do it smarter" section offers shortcuts that reduce time and effort, and help direct the analysis to specific customer needs.

A final section on real-world situations helps the reader begin to see how the thoughtful and appropriate application of statistics can add significantly to the quality of the testing, development, or manufacturing process.

While this may not be the book that provides "everything you ever wanted to know" about statistical methods, it is a most useful guide, the best I've seen. I expect to wear my copy out.

Reader Interest Survey
Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 183 Medium 184 High 185

Completely automated assembly

David K. Kahaner

US Office of Naval

Research, Far East

kahaner@cs.titech.ac.jp

Now that automating production lines has become commonplace, the next logical step is to completely automate assembly lines. Japan's semiconductor industry is aggressively pursuing this trend, as are segments of its household electrical goods industry. Small- and medium-size assembly lines and those where many models are assembled in small lots have yet to implement full automation. But even these are feeling the pressure to climb on the bandwagon.

Assembly lines that have already begun to implement automation, moreover, find themselves in various stages according to the degree of automation they employ, their flexibility, and the feedback they give to product design. Lines where the assembly of existing products has been converted from hand assembly to robot assembly we call "first-generation" automated lines. Those that are automated to the extent that some product design changes can be made we refer to as "second-generation" lines. Lines having automation features that impact broadly on product design then are "third-generation."

All assembly lines, no matter what their generation, are subject to a great many limitations and conditions when they are built. Here we examine the progress made by a range of leading Japanese industries—makers of everything from caterpillars to computers—to guage current progress toward fully automated assembly. Included in our survey are automated assembly lines at Daikin Industries, Gunma Nippon Electric, and Seiko Epson. Of these lines, only Epson's printer assembly line had been previously automated; until very recently the other lines all employed manual assembly.

To get a better feel for these trends, we will look at the developments at one of these companies, Gunma Nippon Electric, to see what a fully automated assembly line entails. The basis for this analysis comes from an article in *Nikkei Mechanical* (June 29, 1992).

Automation-oriented design techniques

Much of the credit for the traditional cost competitiveness and high quality of Japanese manufactured goods belongs to the skillful and pervasive automation of its production lines. More recently, automation there has become a critical component in dealing with labor shortages, demands for improved work environments, and the need to implement companywide computer-integrated manufacturing (CIM).

Discussion currently focuses on automating the assembly line. This emphasis is quite distinct from automating fabrication lines—a simpler case where systems need only be built to automate the conveyance of work to numerically controlled machine tools and the like. In automating assembly lines, robot hands and jigs must be adjusted to handle each part, greatly complicating parts feeding and line control.

Now that industry has made so many strides with factory automation, equipment, production control, and computers, building automated assembly lines no longer seems unusual. Until now, however, either high-volume production operations, such as for electrical goods, or products having relatively simple structures, have accounted for most of this assembly automation. These applications have not, however, spread through the entire manufacturing industry, as has been the case with fabrication line automation.

Three technological advances. A broad range of applications has advanced dramatically by automating their assembly lines. Included are

0272-1732/93/0600-0088$03.00 © 1993 IEEE

everything from commercial outdoor air conditioners (Daikin) and hydraulic shovels (New Caterpillar Mitsubishi) to personal computers (Gunma Nippon Electric), and computer printers (Seiko Epson).

In these lines, products that come in many different models must be assembled in small- and medium-size lots and require high-precision assembly, making automation difficult. Labor shortages and demands to implement computer-integrated manufacturing have increased the pressure to automate these lines.

Assembly automation applications have broadened, however, with the recent advances in production technology, including robots and sensors, component feeding, line control, and production management. The new printer assembly line that Seiko Epson began operating in April 1992 provides one good example of these advances.

Seiko Epson had successfully automated its assembly operations in 1984. To take advantage of subsequent developments in automated assembly technology, the company needed to refurbish its line, a decision that also greatly influenced the design of the company's new printer models.

What, specifically, are these advances in automated assembly technology? A survey of the latest automated lines in Japan noted three major areas of development:

- Positioning technology, which is fundamental to robotics,
- High-flexibility line construction technology that can cope with the mixed-flow production of multiple product models, and
- Product design technology aimed at assembly automation.

Multipurpose use of visual sensors. Common to all assembly automation is the problem of positioning. Automation would be a relatively simple procedure if parts could be neatly lined up and carried along, but the size and shape of parts often makes this impossible. Related problems include the need to reposition parts such as the large metal plates used in hydraulic shovels from which arise plate fabrication irregularities. Deformations in parts as with personal computer printed circuit boards can also create difficulties. Frequently, an assembly point will be out of position even though the end or edge of a part is properly placed.

Automation would be relatively simple if parts could be neatly lined up and carried along.

The first step in positioning is to develop jigs that match the parts. After taking up a part, a robot first must place it on or in a jig before assembly can commence. Then the robot must grip the part again. The jig determines the position and attitude of the part. With large metal plates, positioning can only take place after special jigs have been developed to hold the plates in the proper position for assembly.

The shape of a part can make positioning with a jig difficult. Sensors sometimes must be used to correct the movement of the robot, but accurate sensor detection is often very difficult. Printboard warping provides a typical example. After running into problems with printboard warping, Gunma Nippon developed a triangular measuring optical sensor for use in detecting the height of three points on a printboard. A two-dimensionally curving warp can be represented by three points, so the company had to come up with innovations in measurement point selection and interpolation techniques.

Visual sensor applications have now become somewhat standard. Visual sensors can detect the positions of a great many parts and are effective in implementing mixed-flow production operations. In addition to detecting positions, these sensors can also detect the shapes of parts, making them most useful in product quality control.

To be sure, the positional correction precision of a visual sensor is not terrific. When we factor the mechanical error of the robot together with the limitations of image processing resolution, errors of several hundred microns can and do develop. More innovation for high-precision assembly soon becomes necessary.

In Daikin's outdoor air conditioner unit assembly line, visual sensors combine with remote center compliance (RCC) mechanisms to perform high-precision nesting operations with compressor parts. The compressors used by Daikin are scroll-type compressors that require approximately 10-μm precision in operations such as nesting crankshafts into bearings. Visual sensors alone do not afford this degree of precision, so RCC mechanisms attached to the robot hands must absorb the remaining error. The RCC mechanism contains an internal spring mechanism, enabling it to search for accurate nesting positions by the deflection of the spring from work forces.

Movable jigs and sensors simplify production control. The assembly lines at New Caterpillar Mitsubishi and Daikin are designed to cope with the demands of multiple-model, mixed-flow production. To increase flexibility, the assembly operations there use movable jigs to implement controls that accord with part shape. Visual sensors also play an important role.

In a mixed-flow assembly line, information on product models must be controlled so that the line operation

can be adapted to match the model moving on the line. Basically a computer working over a LAN handles this information control, but simpler and more reliable methods are needed.

Recognizing this need, Daikin has equipped its pallets with memory cards to achieve "object-information integration." The memory cards contain model information such as parts dimensions, detection standards, and other vital data that can be read at each assembly stage. This way, the operation can be changed easily to accord with the model.

Daikin employs its own methods for line control and parts supply. It was a sequence-synchronized production method for ordering the supply of parts assembled in the main work flow, and attach numbers to the work to indicate production order. Parts then can be ordered from the automated warehouse based on these numbers.

Focus on modular design. If we approach assembly automation merely from the perspective of production technology, however, problems involving equipment costs and reliability soon arise. Consequently, it has become standard practice to reevaluate product designs and to implement design features compatible with automated assembly operations. We can then simplify assembly and enhance operational reliability by, for example, orienting all the assembly steps in one direction, or employing connection techniques amenable to automation.

To maximize the effectiveness of these measures, we would prefer to have "new lines for new models." That way we could develop parts concurrently with line construction. Investment efficiency and problems arising from the product model change cycle often force us, however, to automate a line for an existing model.

Looking at recent design trends, the modular approach adopted by Daikin on its outdoor air conditioner units is noteworthy. For design purposes, product structures are divided into a number of modules. Each module is assembled on a subline, and the assembly operations that are not amenable to automation are concentrated in the final assembly line. Automation rates in the total assembly process rise easily because each module is designed for compatibilility with automated assembly. Even automobile manufacturers are attempting to modularize their products so that they can automate the final assembly process.

These product design techniques should change as assembly automation technology continues to progress. The printer line at Seiko Epson provides an extreme example. For some time, Sieko Epson has tried to implement unidirectional parts assembly to facilitate assembly automation. Unidirectional assembly, however, translates into complex parts shapes and higher fabrication costs—the total manufacturing cost escalates even though assembly costs are low.

Investment efficiency and model change cycles often force us to automate lines for existing models.

With robots becoming so highly functional, inserting a part diagonally no longer presents the challenge it once did. The best policy then is to calculate parts fabrication costs and assembly costs to arrive at the most favorable method of assembly. Seiko Epson designed its new printers in the context of total manufacturing cost and constructed its assembly line accordingly.

Improving the automation rate while developing ways to handle multiple-module, mixed-flow production becomes the next task. The mixed-flow production approach is effective in holding down equipment costs and coping with production fluctuations.

The key to implementing assembly automation is the development of automation-compatible designs. This design-oriented approach is a new concept, but one with great potential. Developing programs to coordinate design and production now takes on increasing urgency.

To get a clearer picture of all that fully automating the assembly line involves, let's take a nuts-and-bolts look at one of these companies.

Automated computer assembly

Gunma Nippon Electric is the main development and production center for personal computers carrying the NEC name. Gunma NEC built an automated assembly and inspection line for desktop PCs in November 1991. These computers represent more than 10 million users. Some 1.17 million units were produced in 1991. But even at NEC, which has more than half the domestic share in this market in Japan, computers were assembled manually until 1991.

"We began planning the automated assembly line in 1987," says Masaki Takahashi, manager of the Systems Divsion of the CIM Systems Department. "But the plans were delayed when notebook PCs hit the market and raised the specter of declining desktop demand." Apparently, the biggest obstacle then was return on investment. But what about problems with production technology? Looking at the line, we see that a number of innovations have been implemented.

Moving ahead with production automation. Gunma NEC is working with other parts of NEC to implement computer-integrated manufacturing. They aim to tie production and supply

to market fluctuations. When a product sells well or poorly, the number of PCs produced should rise or fall accordingly.

Toward this end, Gunma NEC wants to shorten what it calls "production multiplier lead time." This refers to the time required to double the number of PCs that the company plans to produce. The company wants to take the time to fix the production plan and, as nearly as possible, set it to the actual number of product days. All the factors involved, from parts procurement to production systems, must be reevaluated to achieve this goal.

Maintaining some degree of overstocking will handle the parts problem. For production systems, excess production capacity should be maintained representing 120 percent of the average number of products shipped. Even these steps, however, cannot fully absorb fluctuations in PC demand. Thus they modified the single-shift production operation, with normal 8-hour shifts, to handle two or three shifts.

However, a problem arises—retaining skilled personnel to work the shifts. Some 15 skilled workers are required to cover a manual assembly line, so 30 must be retained to handle a double shift. But how can this be done when the extra workers are only needed during periods of increased production? The problem will be solved if production can be automated and unmanned production implemented. Gunma NEC is moving ahead with production automation. In 1988, the company automated packaging, and in 1989, they automated the printboard assembly and inspection operations. In late 1991, Gunma NEC built a robot line that automated the final assembly and product inspection stages.

Reducing skilled workers by one third. The robot line has nine assembly operations covering a length of 50 meters, and 12 inspection operations that cover 30 meters, excluding the running test room. Four of the robots employed are six-axis vertical articulated models, eight are three-axis transverse models, one is a two-axis transverse type, and two are horizontal articulated types, for a total of 15 robots. The total investment was approximately 500 million yen.

Of all the operations done on this robot line, only two assembly operations and three inspection operations are performed manually. The two manual assembly operations both involve hooking up cables. The cable assembly operations involve two aspects, namely plugging in the connectors and stringing the cables. Stringing a pliable object like a cable is one of the most difficult jobs for a robot to attempt.

Stringing a pliable object like a cable is one of the most difficult jobs for a robot.

Though this robot line has not completely eliminated manual labor, it has reduced the number of skilled on-site workers required to just five—a 300 percent reduction over the manual line. At the risk of oversimplifying the situation, this automation has made it possible to take the same number of skilled workers from the manual line and spread them over three shifts on the robot line. The start-to-finish tact time has been reduced by 20 seconds, from 77 seconds on the manual line to 57 seconds on the robot line, making it possible for the robot line to turn out 450 PCs in an 8-hour shift.

Components positioned with special jigs. The robot line employs almost the same assembly sequence as does the manual line. The components comprising the PC are very few, including only the baseplate, U-shaped cover, front mask, rear cover, motherboard (on which the processor is mounted), floppy-disk drives, and a few cables. And since the components making up the PC are so few, the sequence in which they are assembled does not need to be changed.

In the assembly operation, the base is first attached to a jig pallet, and the main printed circuit board (motherboard) is installed. The motherboard is screwed into place in the next operation. Next, a chassis for mounting the floppy-disk drive is installed and the drive is mounted in it. A cable between the drive and motherboard is connected by hand. The expansion cage is then built in, and screwed to the floppy-disk drive chassis and expansion cage. A cable is then manually connected to the expansion cage, and the front mask is attached at the same time. The rear and top covers are then attached and screwed into place to complete the assembly. Note that all component positioning occurs simultaneously before any are mounted. The components are not small enough to be supplied by a vibrating parts feeder, so they are placed behind the robots on a pallet.

The robots take components from the pallet and place them in a special positioning jig. The jig positions the components with air-pressure cylinders. From the standpoint of tact time, the reasons for adopting a disadvantageous positioning method are as follows. Using a pallet that can carry the components and perfectly position them involves prohibitive pallet fabrication costs. Placing the components on the pallet then would also be problematic, adding to component costs. Also, most of the components used in a PC are fabricated from sheet metal. Compared to machined parts, the precision of these fabricated units is poor, so automated assembly cannot be done if the positioning is sloppy.

Handling printboard warp. Positioning components with jigs does not solve all the problems. Screw-hole

positioning precision and component warping difficulties remain, for instance. To deal with such problems, one can either measure the screw-hole position or warp and adjust the movement of the robot accordingly, or one can implement more exacting component precision.

The measurement approach requires longer tact times. Using more exacting precision requires higher component costs. These problems led Gunma NEC to reexamine the criteria for all of its components. For a good example, let's look at what they did with the motherboard, a big square printboard measuring 305 mm^2. Conventionally, hole positions had been marked sequentially from one edge of the board, which produces decreasing precision as the distance from the edge to the hole increases.

In securing a printboard, the important consideration is the relative position of one hole to another, much more so than the relative position between a hole and the edge. This being so, beginning with the PC9801 FA series that went on sale January 1992, Gunma NEC changed the design of the printboards to indicate the positions of holes in terms of one standard hole located near the centerline, eliminating the need to measure hole positions.

Tinkering with the standard positions, however, would not resolve the problem of motherboard warping. Almost all the mounting and soldering of electronic components to the printboards has now been automated. The soldering involves wetting the printboards with molten solder; the heat from this process unavoidably produces printboard warping. Since the motherboard is so large, a small angle of warp can produce displacements measured in millimeters.

For this reason, when printboards are positioned at Gunma NEC, the printboard warp is measured with optical sensors using triangulation. Since there is no time to take measurements over the entire surface of the board, three sensors measure the warp at three points simultaneously. For this reason, mistakes are sometimes made in detecting warp. After the motherboard has been installed, it is inspected to see whether or not it is in the right position.

Manual assembly also possible. Unfortunately, a robot line is longer than a manual assembly line. At Gunma NEC, the conventional manual assembly line is 40 meters long, divided roughly equally between assembly and inspection stages. The robot line, however, is 80 meters long, with 50 meters for the assembly stages and 30 meters for the inspection stages. In general, when production is roboticized, the line becomes longer largely because robots have a hard time doing more than one thing. At Gumna NEC, each robot is set up to install two components. Even so, the robot line is twice the length of the manual line in the interest of maintenance. "We allowed plenty of extra space so that it would be easy to perform maintenance on it and revert to manual assembly in the event of a breakdown," says T. Ono, manager of the Production Technical Division.

Research has already begun on ways to automate the assembly of notebook computers.

Easier to add components. Their limited adaptability when models change also creates problems for automated robot lines. A robot can assemble all kinds of components if the program that controls the robot's actions is modified and the robot hand is adjusted properly. If the number of components grows, however, a robot line is not very adaptable at all. To keep the tact time short, more robots must be installed, an approach that is both expensive and time consuming. If the tact time can be lengthened, the problem remains of how to supply the components when the number of components assembled by each robot increases. The robot line at Gunma NEC is built so that the number of components assembled by any one robot can easily increase, making the line highly adaptable to production model changes.

Studies are underway to find ways to eliminate the cables that must be manually hooked up. If successful, this research should make it possible to achieve completely unmanned automation of assembly lines. Research has already begun on ways to automate the assembly of notebook computers, which are much smaller than the desktop PCs and hence much harder to assemble automatically. Gunma NEC intends to employ robots in ways that will make it possible to adjust its production volume to demand vicissitudes.

[David Kahaner is on assignment with the US Office of Naval Research, Far East. His comments are his own; they do not express any official policy.]

New Products

Send announcements of new microcomputer and microprocessor products to Managing Editor, IEEE Micro, PO Box 3014, Los Alamitos, CA 90720-1264.

Joe Hootman

University of North Dakota

CAD tools

VHDL package speeds FPGA design

Release 1.21 of the Complete Optimization/Retargeting Environment (CORE) offers FPGA designers a top-down, three-step design approach based on the VHSIC Hardware Description Language (VHDL). Users enter designs in VHDL, then CORE translates and optimizes them for the target FPGA device. CORE can be seamlessly integrated with IEEE 1076-compliant simulators, along with VHDL-producing systems such as i-Logix, Vista, and Ascent.

This release includes specialized optimization techniques to support Actel Act 3 and Altera Max 5000/7000 series devices. CORE has also been ported to the Hewlett-Packard 700 series hardware platform and the Motif Windowing System. *Exemplar Logic; from $8,000 (CORE pre-seat cost); upgrades free with maintenance contracts.*

Reader Service No. 10

Module supports XC4000s

An FPGA Foundry addition supporting Xilinx XC4000 devices lets designers use a single tool set and take advantage of the Timing Wizard module to set timing parameters and operating frequencies at the beginning of design. FPGA Foundry also works with other FPGA vendors and architectures and integrates into existing CAE environments. The XC4000 release includes fast carry logic, wide edge decoders, RAM, partially or fully placed and routed hard macros, guide files, and a graphical logic block editor. *NeoCAD; from $4,995; $4,500 (PC upgrades).*

Reader Service No. 11

FPGA system targets PALs

PAL users can convert to FPGA design methodologies with Designer, a "purchase-once" FPGA design system that supports the company's devices under 2,500 gates. Designer includes the Action Logic System and choice of design kits for PC 386 and 486 hardware platforms. The three design kit configurations support OrCAD, Viewlogic, and EDIF netlist interfaces and include device macro lines and simulation models. For designs needing higher device densities, Designer Advantage accommodates Act 1/2/3 devices up to 10,000 gates and Sun-4, Sparc, Sparc 2, 386/486, and HP series 700 platforms. *Actel; $995 (Designer). $495 (Designer Advantage, current users).*

Reader Service No. 12

Design Center with PSpice

Available for the Quadra, Powerbook, Mac IIvx, Performa, and Macintosh workstations with math coprocessors and 2-Mbyte RAMs is a CAD/CAE system called Design Center. With this design environment users can simulate analog, digital, mixed analog/digital circuits with PSpice at all levels of the design process and analyze graphical waveforms.

Design Center also supports the HP Apollo 9000 Series 700 workstation, which, according to the company, permits circuit simulations to finish 12 times faster than on a 386/33-MHz PC and twice as fast as on a Sun Sparc 2. *MicroSim; $4,950 (Macintosh version), $17,900 (Series 700 version).*

Reader Service No. 13

FPGA, PLD software combined

The XACT Base Development System combines FPGA and existing programmable logic development software in support of 3,000-gate devices. For FPGAs, the software includes an interface to compile OrCAD or Viewlogic design and offers incremental FPGA design so engineers can quickly make changes. An XDelay static timing calculator, download software, and

0272-1732/93/0600-0093$03.00 © 1993 IEEE

parallel download cable help verify designs. For ELPDs, the system adds Palasm-compatible equation files for design use. *Xilinx.*

Reader Service No. 14

Embedded RISCs to access VxWorks

Designers of high-performance embedded applications based on the AMD 29000 processor will soon be able to access the VxWorks operating system and development tool suite for embedded control. VxWorks lets users develop and execute complex real-time and embedded applications with a Unix cross-development package that networks designs, testing, and debugging tools with target hardware. *Wind River Systems; 4Q93 availability.*

Reader Service No. 15

Tool aids three-layer routing

Microroute, an ASCII-format placement-and-routing solution for three-layer metal, mixed block and cell designs lets users choose either automated or manual approaches. A global router handles corners and intersections across the entire design, while an N-layer detailed maze router produces dense results in specific areas. Designers can create files by writing directly from their system or through EDIF and GDSII Stream interfaces. *Mentor Graphics.*

Reader Service No. 16

IEEE P1284 ECP design kit

System designers wishing to design motherboards and add-in cards to support the IEEE P1284 Extended Capabilities Port protocol can use the ECP/EPP Super I/O Design Kit. ECP provides a high-speed bidirectional port that is backward-compatible with existing cables and connectors. The port should improve the performance and ease of use of parallel peripherals. The kit provides documentation, software, schematics, and a demonstration board. *Standard Microsystems.*

Reader Service No. 17

MCM testing, diagnosis

Multichip module and PCB manufacturers can test and diagnose products with the MCM Probing System, which merges advanced station control and CAD navigation software with the Micromanipulator precision probe placement platform. The IDE-compatible, touch-sensitive probing system allows several probes to be activated simultaneously. It displays or stores real-time waveforms from sampling oscilloscopes and logic analyzers within the DSO and XLA Tool windows using IEEE 488. Users can automatically track full logging of the diagnostic session while accessing interactive control of features via mouse-driven, pop-up menus. The modular system provides automated operation drawing from either IC- or PCB-based design and layout data. *Schlumberger Technologies, ATE Division; from $95,000.*

Reader Service No. 18

Desktops can access CADAM AEC

Engineers with PS/2s can access CADAM AEC tools to design facilities from preliminary layout through the construction phase by adding the Personal/370 Coprocessor. The turnkey system uses a mainframe VM system card that can be inserted into a PS/2 running OS/2. Since it runs host CADAM V3R2MO, this platform has full functional and data compatibility with a mainframe seat; no retraining is required. *Integrated Systems Technologies.*

Reader Service No. 19

DSP systems

Generate C code with Windows

Version 1.0 of the Hypersignal-Windows Block Diagram object-oriented simulation program for signal processing lets users generate C code to run algorithms outside Block Diagram. Block Diagram implements a DSP design by proving the algorithm, debugging it, and setting up what-if situations in a visual environment. The code generator then produces C code for the algorithm, which may be compiled with the DSP chip manufacturer's C compiler. This lets the program run in real time on a DSP. *Hyperception; $2,995 (code generator), $1,995 (Block Diagram).*

Reader Service No. 20

Integrate DSPs/gate arrays

Beginning a new line of standard DSPs embedded in gate arrays is the 100-/144-pin thin SQFP TEC320C25A. System designers can use the chip to develop customized DSP solutions that get to market quickly. The 15K-gate, 60-MHz, 15-MIPS TEC320C25A integrates two standard, high-volume devices onto one chip as an enhanced version of the 16-bit, fixed-point TMS320C25 and an array from the 0.8-micron TGC1000s. Designers can use the TGC1000 library of gate arrays to integrate various logic functions with the DSP core. *Texas Instruments.*

Reader Service No. 21

PC voice recognition

Designed to increase personal productivity by adding a voice command interface tied to keyboard and mouse macros, Voice Blaster system runs on Intel-based personal computers using DOS or Windows 3.1. The voice recognition system includes a toolbox of revised programs and utilities for recording, editing, and playback, as well as voice annotation software that adds a user's own recorded messages to documents. A high-fidelity headset with microphone and speaker connects to a user's computer via the parallel port. Though fully functional on a 286 with 640-Kbyte RAM, in the presence of EMS memory, Voice Blaster software will automatically load as much of itself as possible in high memory. *Covox; $119.95.*

Reader Service No. 22

Multimedia PC audio decoding

Two single-chip audio coder/decoders address the needs of multimedia

personal computers, providing stereo, 16-bit audio. The 68-pin PLCC-packaged CS4248 and CS4231 codecs are pin-compatible with the Analog Devices AD1848 and come with Windows-compatible drivers.

The CS4248 ADC/DAC uses proprietary delta-sigma conversion techniques to code and decode audio signals, making CD-quality sound available. It includes an 8-bit parallel ISA/EISA interface, analog mixers, antialiasing and reconstruction filters, and simultaneous capture and playback capabilities. The enhanced CS4231 version offers 4-to-1 adaptive differential pulse code modulation compression/decompression and supports different data formats. *Crystal Semiconductor; from $35 each (1,000s).*

Reader Service No. 23

ADCs speed at 1-Msamples/s

LTC1273, LTC1275, and LTC1276 300-Ksample/s analog-to-digital converters contain a precision reference, a high-speed sample and hold, and an internal clock. Typical signal-to-noise distortion on the 24-pin narrow DIP or 24-lead SOIC chips is 72 dB for 10-kHz inputs and 70 dB for 100-kHz. The LTC 1273 runs on one 5V supply and converts 0V to 5V inputs. The LTC1275 and LTC1276 run on ±5V and convert ±2.5V and ±5V inputs.

The LTC1196 and LTC1198 1-Msample/s, 8-bit ADCs come in S0-8 surface-mount chips and offer 600-ns conversions. The switched-capacitor, successive-approximation chips include 100-ns sample and hold on chip; both operate from 2.7V to 6V power supplies. *Linear Technology; from $14.09 (300-Ksample versions, 100s), from $2.37 (1-Msample versions, 1,000s).*

Reader Service No. 24

Mac-based data acquisition

The 50-Ksample/s MacScope system provides hardware and software for educational, research, and industrial applications where multichannel data acquisition is required. The System 7-compatible product running on a Macintosh Plus, SE, or II also provides data analysis such as Fourier transfoms. Features include a mouse, pull-down menus, scroll bars, and data file storage with screen-printing options. *World Precision Instruments.*

Reader Service No. 25

Acquire data with Windows DDE system

A recent version of Snap-Master for Windows 3.1 lets engineers and scientists acquire, display, analyze, and output data with Dynamic Data Exchange support. Snap-Master version 2.0 integrates sensors, transducers, and signal conditioning. Features include context-sensitive on-line help, zooming and panning for large files, multiple cursors, and event markers.

Users can transfer data to a spreadsheet for real-time trend analysis and report generation while Snap-Master acquires information in the background. Version 2.0 requires a PC or PS/2 and a 4-Mbyte memory and comes in three stand-alone modules that also work as an integrated package. *HEM Data; from $495 (modules), $1,985 (package).*

Reader Service No. 26

VME boards provide DSP subsystems

According to the company, its 200-MFlops/1-Gops 6U VME boards based on the TMS320C40 DSP let embedded systems designers reduce development time by 50 percent. Both CV2 and CV4 boards combine DSP, array processing, and parallel processing capabilities of the C40 with I/O and standard interfaces. Debuggers, libraries, compilers, assemblers/linkers, and other software tools complete the systems.

Each board can be used with TIM-40 modules to optimize designs and can be configured with eight processors, large DRAM arrays or fast SRAM arrays, and special-purpose modules such as video capture and SCSI. *Spectrum Signal Processing; from $9,100.*

Reader Service No. 27

Voice processor boasts open architecture

The BT-IV digital voice processor delivers high-fidelity speech, built-in redundancy, and digital-controlled components for application program control provided by micro, mini, or mainframe computers. The tower or rack-mount BT-IV connects to value-added services such as ISDN and world standards. A switch-host feature enhances fail-safe applications. *Perception Technology; $1,500 per channel.*

Reader Service No. 28

Communications/displays

Ethernet link added to multiscreen display

The Media Wall multiscreen display system for control room and simulator applications features an Ethernet link to Sun, SGI, HP, DEC, IBM, and other workstations. In normal room lighting, Media Wall displays graphical or photographic information in an array of 144 monitors or projectors controlled by one computer. The displays can be placed in one line or circle, a rectangular matrix, or other shapes for specific requirements. *RGB Spectrum.*

Reader Service No. 29

VGA monitor survives in harsh areas

Designed for extreme conditions of temperature, humidity, dirt, shock, and vibration, a slim flat-panel line of VGA monitors displays graphics in active matrix color LCD or monochrome EL screens. Available with infrared touch systems and touch mouse features, the Seal Touch monitors weigh about 25 pounds and come in stand-alone or OEM module versions. *Lucas Deeco; from $3,650; delivery 60 days ARO.*

Reader Service No. 30

X terminal promises 1,200×1,024 resolution

A RISC-based, color flat-panel X Terminal, the XfaceC, features 70,000-

Xstone performance and 1,280×1,024 resolution. The 25-MHz system designed to work with most platforms and in multiple-host environments offers 256 simultaneous colors on a 13-in. LCD screen and an X server accelerator to deliver fast screen updates and offload bitblt, fill, and arc primitives to hardware. *Japan Computer Corporation; $10,000.*

Reader Service No. 31

Communications programs added

Three communications packages support DOS, Microsoft Windows, and Windows NT.

Version 3.4 of DynaComm/Elite for Windows-based PC-to-mainframe connectivity offers concurrent 3270 and LU6.2/APPC communications. Version 2.2 of 3270/Elite Plus is a multisession, DOS-based, PC-to-mainframe connector for SNA LU2 environments.

The third package, the 3270 Emulator, connects over IEEE 802.2 networks and supports one host display session plus copy and paste functions. The simplified 3270 PC-to-mainframe communications program ships with each copy of the 32-bit Windows NT operating system for PCs. *Network Software Associates; $395 (Elite packages); corporate and network licenses available.*

Reader Service No. 32

Displays use PCMCIA memory

Max/it displays come with Options/Open to use PCMCIA flash memory technology in general-purpose alphanumeric terminals. Two protocol-specific/ANSI emulation terminals, the Max28 and the Max1120, are designed for the Uniscope market and require one PCMCIA card per site. The Max10BT includes a 10BaseT and an AUI Ethernet port for plug-in LAN connection and works with TCP/IP or Novell Netware protocols. The Max6 multisession ASCII/ANSI/graphics terminal completes the family. *Link Technologies; from $649.*

Reader Service No. 33

ATM hub delivers LANs/WANs

An Apex Asynchronous Transfer-Mode switching hub for corporate networks supplies both pure cell switching and adaptation switching interfaces for non-ATM traffic in one platform. ATM supports very high transmission speeds for integrated data, voice, and video. Apex interconnects LAN hubs with ATM or Ethernet interfaces; transports circuit-switched data, voice, and video; switches frame relay and X.25 traffic; and transmits HDLC and SNA/SDLC-framed information within one backbone network. *General DataComm; from $50,000 to $125,000 (typical system configurations).*

Reader Service No. 34

IRMAs support Windows NT, OS/2

IRMA Workstations now support Microsoft Windows NT and OS/2 2.0 Presentation Manager with a 32-bit host access communications package that integrates the IBM SNA network. Both versions can access DFT, SDLC, X.25, and IEEE 802.2 environments over token ring or Ethernet. They emulate the IBM 3270, Logical Unit 6.2, LU0, and LUA Advanced Program-to-Program Communications. Features include a keyboard editor and QuickPad for frequently used keys. The NT package can process 10 concurrent 3270 sessions. *Digital Communications Associates; $495 each version.*

Reader Service No. 35

Multiplatform X.5 servers

A line of X server software solutions based on release 5 of the X Window System from MIT offers Unix connection to Apple Macintosh, NextStep, and MS Windows computers. Known as both X11R5 and eXodus 5.0, the multiplatform products support X font formats, networked X font servers, and the enhanced DECwindows, Sun Open Windows, and Motif. *White Pine; $295 each (Macintosh version), $349 (NextStep); $449 (MS Windows).*

Reader Service No. 36

Simplify ASIC design

Eight FSB cells for T1/E1 applications in PCM multiplexing, switching, and transmission systems are blocks of application-specific logic that may be embedded into an ASIC chip. After being surrounded with user-specific logic, the cells provide a customized solution for a system design problem. By combining several functions and up to several thousand gates, system designers can quickly implement specific communication functions. *VLSI Technology.*

Reader Service No. 37

Wireless LAN connects PC shoppers

Small-business customers requiring simple, powerful computing solutions can purchase a wireless peer-to-peer LAN system called Advantage! Net. With built-in wireless communications and preinstalled application software, the hard-wired network alternative uses Windows for Workgroups linked to a wireless RangeLAN adapter from Proxim. A 25-MHz, 486SX-based, 4-Mbyte desktop and an 8-Mbyte, 66-MHz, 486DX2 minitower with 120-Mbyte tape backup make up Advantage! Net. Sold nationwide at 900 retail locations. *AST Research; under $2,000 (486SX/25), $3,500 (486DX2/66).*

Reader Service No. 38

Handheld protocol analyzer for notebooks

A PC-based protocol analyzer line communicates with a PC via a standard 4-bit or 8-bit parallel port and does not require a card slot. Designed to work with the low to medium WAN segment of data communication testers, the handheld Feline PS2002 ParaScope is housed in a 6.22×3.74×2.17-inch molded plastic case. It offers 19.2-Kbps RS-232 data monitoring and analysis. A PS6002 64 works with applications requiring 64K performance, while the PS6145 64M offers 64K performance with integral interfaces for RS-232, X.21,

READER SERVICE CARD

IEEE MICRO

June 1993 issue (card void after December 1993)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest
(Add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles and departments):

150	151	152	180	181	182
153	154	155	183	184	185
156	157	158	186	187	188
159	160	161	189	190	191
162	163	164	192	193	194
165	166	167	195	196	197
168	169	170	198	199	200
171	172	173	201	202	203
174	175	176	204	205	206
177	178	179	207	208	209

Product Information
(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	

READER SERVICE CARD

IEEE MICRO

June 1993 issue (card void after December 1993)

Name _____

Title _____

Company _____

Address _____

City _____ State _____ Zip _____

Country _____ Phone(_____) _____

Reader Interest
(Add comments on the back)

Readers,
Indicate your interest in articles and departments **by circling the appropriate number** (shown on the last page of articles and departments):

150	151	152	180	181	182
153	154	155	183	184	185
156	157	158	186	187	188
159	160	161	189	190	191
162	163	164	192	193	194
165	166	167	195	196	197
168	169	170	198	199	200
171	172	173	201	202	203
174	175	176	204	205	206
177	178	179	207	208	209

Product Information
(Circle the numbers to receive product information)

1	21	41	61	81	101	121	141
2	22	42	62	82	102	122	142
3	23	43	63	83	103	123	143
4	24	44	64	84	104	124	144
5	25	45	65	85	105	125	145
6	26	46	66	86	106	126	146
7	27	47	67	87	107	127	147
8	28	48	68	88	108	128	148
9	29	49	69	89	109	129	149
10	30	50	70	90	110	130	
11	31	51	71	91	111	131	
12	32	52	72	92	112	132	
13	33	53	73	93	113	133	
14	34	54	74	94	114	134	
15	35	55	75	95	115	135	
16	36	56	76	96	116	136	
17	37	57	77	97	117	137	
18	38	58	78	98	118	138	
19	39	59	79	99	119	139	
20	40	60	80	100	120	140	

SUBSCRIPTION CARD

SUBSCRIBE TO IEEE MICRO

All the facts about today's chips and systems

❑ **YES**, sign me up!

If you are a member of the Computer Society or any other IEEE society, pay the member rate of only $23 for a year's subscription (six issues).

Society: _____

IEEE membership no: _____

Society members: Subscriptions are annualized. For orders submitted March through August, pay half the full-year rate ($11.50) for three bimonthly issues.

❑ **YES**, sign me up!

If you are a member of ACM, ACS, BCS, IEE (UK), IEEE but not a member of an IEEE society), IECEJ, IPSJ, NSPE, SCS, or other professional society, pay the sister-society rate of only $42 for a year's subscription (six issues).

Organization: _____ Membership no: _____

For information on airmail option, send fax request to (714) 821-4010.

_____
Full Signature Date

_____
Name

_____
Street

_____
City

_____
State/Country ZIP/Postal Code

❑ Payment enclosed *Residents of CA, DC, Canada, and Belgium add applicable sales tax.*

❑ Charge to ❑ Visa ❑ MasterCard ❑ American Express

Charge-card number

_____

Expiration date

____ ____
Month *Year*

Prices valid through 12/31/93

693 MICRO

Charge orders also taken by phone:
(714) 821-8380 8 a.m. to 5 p.m. Pacific time
Circulation Dept.
10662 Los Vaqueros Circ., PO Box 3014
Los Alamitos, CA 90720-1264

Editorial comments

I liked: _____

_____

_____

I disliked: _____

_____

_____

I would like to see: _____

_____

_____

Reviewers Needed. *If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.*

For reader service inquiries, see other side.

PLACE POSTAGE HERE

PO Box is for reader service cards only.

IEEE Micro
PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA

Editorial comments

I liked: _____

_____

_____

I disliked: _____

_____

_____

I would like to see: _____

_____

_____

Reviewers Needed. *If interested, send professional data to Dante Del Corso, Dipartimento di Elettronica, Politecnico di Torino, C.so Duca degli Abruzzi, 24, 10129, Torino, Italy.*

For reader service inquiries, see other side.

PLACE POSTAGE HERE

PO Box is for reader service cards only.

IEEE Micro
PO BOX 16508
NORTH HOLLYWOOD CA 91615-6508
USA

NO POSTAGE NECESSARY IF MAILED IN THE UNITED STATES

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO. 38 LOS ALAMITOS, CA

POSTAGE WILL BE PAID BY ADDRESSEE

IEEE COMPUTER SOCIETY
PO BOX 3014
LOS ALAMITOS CA 90720-9804
USA

V.35/36, V.10, V.11, and RS-449. Both 64 and 64M ParaScopes connect to ISDN lines via company interface pods. *Frederick Engineering; $1,695.*

Reader Service No. 39

No extra power needed for converter

Model 263, an RS-232/RS-422 interface converter, operates under power from the signals applied to the RS-232 interface, provides full-duplex operations for 19.2-Kbps transmit and receive data signals. The 2×2.75×0.75-inch RS-422 signal interface simultaneously serves both screw terminals and an RJ-11 connector. For DTE or DCE configurations, users activate a switch that reverses pins 2 and 3 of the RS-232 connector. *Telebyte Technology; $89, quantity discounts available.*

Reader Service No. 40

Windows IRMAs announced

IWW 2.1 and IWD 2.0 PC-to-mainframe software includes support for TN3270 over TCP/IP and NetWare for SAA plus enhanced productivity features. IRMA Workstation for Windows 2.1 includes a Quickbar 3270 feature for easy access to Windows and 3270 functions such as session activate/deactivate, file transfer, and a graphical keyboard editor. IRMA Workstation for DOS 2.0 also provides the editor with remote diagnostic support for company-customer interaction. *DCA; $495 (IWW 2.1), $425 (IWD 2.0).*

Reader Service No. 41

NS/DOS product restores files

Upstream/PC 2.1.0 provides IBM's Networking Services/DOS with unattended backup and restoration of critical PC and LAN data to and from an MVS mainframe. Operating in both DOS and Windows environments, NS/DOS lets workstations use Advanced Peer-to-Peer Networking technology, while low-memory workstations on the network use APPC LU 6.2 services.

Upstream also handles disaster recovery, addresses issues of client/server or distributed applications, and lets data in VSAM clusters be archived to tape for long-term storage. *Enterprise Data.*

Reader Service No. 42

T1/E1 chips send digital data

Two T1/E1 line interface unit chips can be used in PCM multiplexing, switching, and transmission systems. Designated the VP14335 and VP14574, the 28-pin PLCC and DIP chips offer a single-chip solution for synthesizing DSX-1 and CCITT G.703 pulses. The VP14335 provides jitter attenuation on the transmitting side of the signal while the VP14574 provides the same on the receiving side. *VLSI Technology; $9.70 (10,000s).*

Reader Service No. 43

Server runs on nine platforms

Support for Univel's Unixware, Data General Aviion, Interactive, HP/US, and Acer/Altos platforms has been added to a QX15 ASCII/ANSI terminal that runs X Windows applications. The 68000-based text terminal with GUI and mouse port already supports SCO Unix and Open Desktop, RS/6000 AIX, and Sun environments. To prevent screen flicker, the QX15 provides a 78-Hz refresh rate on its 14-inch, flat, white/green/amber phosphor CRT. *Qume Peripherals; $699; X server software available for $200 one-time site license.*

Reader Service No. 44

Modem set corrects errors

Manufacturers can build a complete modem with full data integrity and enhanced features in an area smaller than one half the size of a credit card with the two-chip CL-MD9624EC2. This data/fax/voice modem device set provides MNP4 and V.42 protocols for error correction and MNP5 and V.42bis protocols for data compression at 2,400-bps transfer rates and 9,600-bps facsimile transfers. On-chip communications firmware eliminates the need for software development and debugging. *Cirrus Logic; $23 (10,000s).*

Reader Service No. 45

VME64 adapters, WAN controller announced

The PT-VME600 FDDI and PT-VME430/432 SCSI-2 adapters join the PT-VME340 VMEbus controller in aiding network communications.

The PT-VME600 Fiber Distributed Data Interface node adapter for VME64 systems makes use of National Semiconductor's two-chip set to provide a 100-Mbps data transmission path between nodes. A dual-attached version of the PT-VME600 adapter supports front-end network applications such as image processing, multimedia, and CAD/CAM. It can also serve as a backbone network to tie together the main components of a distributed system.

The PT-VME430/432 SCSI-2 host adapters promise 17-/19-Mbyte/s sustained data transfer rates between the host and storage peripheral devices. Recommended for Unix applications, these adapters achieve 2-µs/data block overhead and support 8-/16-bit SCSI bus widths, with either differential or single-ended operation.

The PT-VME340 four-port WAN controller is a VMEbus Serial I/O module designed for T1/E1++ rates of 10 Mbps. Built around the Zilog 16C32 controller, this interface is based on many of the features and functions of the Z8530 serial communications controller found in current applications. *Performance Technologies; $3,996 (PT-VME600, 100s), $2,570 (/432), $2,236 (/340, 100s).*

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 189 Medium 190 High 191

Product Summary

Joe Hootman
University of North Dakota

Manufacturer	Model	Comments	R.S.#
Chips			
National Semiconductor	DP84910VHG read channel	Third-generation integrated read channel for hard-disk drives provides 50-Mbps performance and advanced power management. A 5V power supply drives the PQFP chip. *$25 (33 Mbps); $30 (50 Mbps) 1,000s.*	80
Philips Semiconductors	TEA1093 telephone set	IC implements a line-powered, "hands-free" telephone set with on-chip supply regulation, microphone and loudspeaker amplifiers, duplex controller with speech and background noise envelope monitors, and channel-switching logic. The 28-pin surface-mount or DIP chip also works with AC-powered equipment. *Hfl 6.00, depending on importing country (10,000s).*	81
Systems			
GammaLink	Isofax 400 gateway	Platform integrates GammaNet fax server software and board to provide inbound and outbound faxing capabilities for X.400 network users. The 9,600-bps system converts on-board text and graphics and provides zero fill for high throughput. Multiple boards can be installed in one PC chassis with multiple sending/receiving lines. *$3,450.*	82
Philips Semiconductors	CDT610/611 CD systems	With a 3-beam CD drive and disc-loading mechanism, plus digital, analog, keyboard, and display electronics, these systems let users design a CD player for use in Midi hi-fi units, in-car entertainment systems, and portable CD-radio cassette units. A mask-programmed microcontroller implements random-play, forward/reverse track searching, and remote control features.	83
Sparcom	Smart Dock connectors	Intelligent docking stations for 512-Kbyte and 1-Mbyte HP 95LXs connect the palmtop with facsimile machines, electronic information services, printers, desktop PCs, and Macintosh computers. Each system includes Data Exchange software for PC or Mac. *From $169.95.*	84
Transtech Parallel Systems	TTM200 memory interface	Module incorporates a 50-MHz i860XP vector processor to provide 400-Mbyte/s sustained data rate and 20-Mbyte memory. TTM200 application development tools, C and Fortran 77 compilers, and symbolic debugger included.	85
VideoLabs	FlexCam camera	A 1/3-in. color CCD camera with two directional stereo microphones mounted on an 18-in. gooseneck arm outputs NTSC video and line-level audio. The integrated system is compatible with most Macintosh and Video for Windows video digitizing boards. The unit's base houses all electronics. *$595.*	86

Manufacturer	Model	Comments	R.S.#
Software			
Andersen Consulting	Windows Client Option, V. 1.2	Foundation Cooperative Processing client/server application lets users create enterprisewide systems for Windows 3.1 and OS/2 Presentation Manager environments that incorporate Windows-based personal computers. At generation time, users can select the Windows radio button to generate Windows clients, and can provide the application with PM and Windows user interfaces.	87
Eyring	Pxrom operating system	Modular, real-time porting environment for Motorola IDP boards based on the M68000 supports the Microtec ANSI C compiler on DEC, HP, PC, and Sun hosts for robotics, industrial controls, and data acquisition applications. Developers can configure PDOS modules for various system architectures and acquire runtime licensing only for the used modules. *From $6,000 plus $175 per license.*	88
Intelligent Systems International	Virtuoso programmer	The Virtual Single Processor Programming system based on the company's API-compatible RTXC/MP real-time kernel can be considered as a microkernel while the lowest layers use nano-kernel technology. Runs on 68HC11, 680X0, 96002, 80X86, T2/T4/T8XX, R3000, and TMS320C30/31/40 systems. *$3,995 (single-processor version); $12,995 (multiprocessors); site developer's license.*	89
Mercury Interactive	WinRunner tester	X-Windows automated tester uses context-sensitive technology to ensure test accuracy and reliability and enable test portability between platforms. DECstation, Sparcstation, HP 9000/700, and RS6000 tool interprets high-level, context-sensitive commands and executes them through the GUI as keystrokes and mouse movements. *$6,000 per license.*	90
National Instruments	PID control packages	Software accesses PID control through graphical or text-based programming tools for Labview for Windows and LabWindows for DOS. PC and Macintosh packages offer P, PI, PD, and PID algorithms; lead/lag compensation; automatic/manual control mode; and error-squared PID. *$295.*	91
Oasys	68K Cross Tool Kit	Development tools for Alpha AXPs include Green Hills C++, C, Fortran, and Pascal cross compilers; the Oasys 68K Cross Assembler/Linker system; and Multi, a window-oriented, source-level debugger. DECstation and Vax systems under Open VMS or Ultrix, most Unix workstations, and PCs running MS Windows can also use the tools.	92
Miscellaneous			
Tadpole Technology	Sparcbook keyboards	European local language keyboards support Sparcbook notebook workstations, enhancing their international Sun Type-4 and Type-5 keyboards. Each integral keyboard includes a mouse key, 12 function keys, and 82 full-size keys.	93

Information for Authors

April 1993

Who we are

IEEE Micro, a bimonthly publication of the IEEE Computer Society, reaches an international audience of microcomputer and microprocessor designers, system integrators, and users. Readers seek to increase their technical knowledge of computers and peripherals; systems, components, and subassemblies; communications, instrumentation, and control equipment; and software.

What we publish

IEEE Micro publishes original works about 5,500 words long (about 20 double-spaced typed pages that include explanatory figures, tables, and programs). These works discuss the design, performance, or application of microcomputer and microprocessor systems. Readers welcome tutorial material, review papers, and discussions of standards. Topic areas include

- systems
- fault tolerance
- languages
- application software
- algorithms
- architecture
- data acquisition
- operating systems
- artificial intelligence
- communications
- hardware/software design and implementation

Submitting your manuscript

Submit six copies of your manuscript and a 50- to 70-word abstract with keywords, your mailing address, phone and fax numbers, and electronic mail address directly to:

Dante Del Corso
Editor in Chief, *IEEE Micro*
Dipartimento di Elettronica
Politecnico di Torino
C.so Duca degli Abruzzi, 24
10129 Torino, Italy
Telephone: + 39 11 564 4044; fax: + 39 11 564 4099
Compmail: d.delcorso; Bitnet: delcorso@itopoli;
Internet: delcorso@polito.it

or

Maurice Yunik
Associate Editor in Chief
Dept. of Electrical Engineering
University of Manitoba
Winnipeg, Manitoba R3T 2N2 Canada
Telephone: (204) 474-8517; fax: (204) 275-0261
Internet: yunik@eeserv.ee.umanitoba.ca

All manuscripts pass through a peer-review process consistent with other professional-level technical publications. This process may take three months, and referees may require revisions to parts of your work. If a manuscript exceeds the specified length, it will be shortened.

Successful contributions avoid the style of transactions and academic journals. They sufficiently introduce the material, place it in context with similar works, describe the practical or potential applications of the material presented, and discuss both pros and cons of the approach. At least 20 percent of the article should be tutorial in nature. Brief literature surveys do not satisfy these requirements.

Upon accepting your manuscript for publication, the Editor in Chief will ask you to supply three copies of any revised draft, plus drawings, photographs, equations, and programs; an electronic version; and biographies and photos of all authors. In addition, you must sign a release transferring copyright to the IEEE (excepting certain key rights retained by the author). Details follow under the Copyright heading.

Submit the hard copies, including illustrations and references or bibliographies, printed on one side only of 8 1/2 × 11-inch paper and double spaced with at least 1 1/2-inch margins. Send an electronic copy on floppy disk or via Compmail or Internet. All electronic files should retain any text-formatting codes you use and identify the formatter used. Refer to the Computer Society's Electronic Submittal Guide for further details. Disks must be Macintosh-compatible or 5.25-inch, IBM PC-compatible, and running DOS Version 2.10 or newer.

For further guidance, contact:

Marie English, Managing Editor, *IEEE Micro*
10662 Los Vaqueros Circle; PO Box 3014
Los Alamitos, CA 90720-1264
Telephone: (714) 821-8380; fax: (714) 821-4010
Internet: m.e.english@compmail.com

Professional editors on the *IEEE Micro* staff thoroughly edit accepted manuscripts. This collaborative process between author and editor results in a concise, well-worded article. Editing covers grammar and punctuation; content (flow, meaning, clarity, directness, organization); and style (conformance to house style).

Copyright

When parts of a manuscript have already been published elsewhere, the author must seek permission from the original publisher. The article will acknowledge the permission; for example, "Section ... and figure ... appeared in .... They are reprinted with permission of the publisher." If IEEE originally published this material, permission is automatically granted, but the original publication must still be cited. Detailed descriptions of author and IEEE rights appear on the IEEE copyright form (yellow sheet).

Manuscript date of receipt

We will publish the date we received the manuscript; just request this when submitting the final version of an accepted article.

Writing tips

Readers welcome clear, accurate articles presented in logical sequence. Let readers know in the first paragraph why your subject is important; give them a reason to continue reading. Define your problem and discuss your solution and any trade-offs. Augment your discussion with examples, tables, diagrams, charts, and photographs to help readers grasp your point. End by putting your topic in perspective and stating any future work plans. Remember, all readers won't be familiar with your specialty; you will have to explain unusual terms or intricate processes.

Readers move swiftly through articles written in the active voice and containing short words, short sentences, and concrete examples. (An active voice example: "This scheme contains two main buses" NOT "Two main buses are contained in this scheme.") Avoid jargon, explain acronyms, and simplify your language. For example, use "to" NOT "for the purpose of" and use "can" NOT "has the capability to." In other words, write the way you talk.

As you can see, magazine style differs from journal and report styles.

References and bibliographies

References substantiate points made in the text or direct readers to other points of view or important works. Do not overdo it, however; most articles need less than 10 citations. They appear in numerical order in the article and in a separate section at the end of the article. Citations in the text appear as Arabic superscripts, for example, Smith.[1] (Use square brackets if your word processor does not allow superscripts.)

Cited sources should be available to the reader; don't include unpublished works. Any abbreviations should follow *IEEE Micro* usage; see a recent issue for examples. When in doubt, spell it out.

You should attempt to provide full bibliographic data as a courtesy to your readers. A complete citation includes author(s); title of article or chapter; title of journal, book, proceedings, or dissertation; publisher's name, city, and state for books and dissertations; complete address for private technical reports; year published; and inclusive page numbers.

Illustrations

Submit photocopies of artwork, rather than originals, for the initial manuscript review. All final illustrations and drawings should be clear and submitted in hard copy (on separate sheets). *IEEE Micro* reproduces your original halftones, machine-made graphs, computer printouts, and electronically produced artwork. Artists will redraw all other art to meet house standards. Photographic prints should have good contrast and be at least 7.5 × 12 cm (3 × 5 inches). Check to see that all artwork is accurate and unambiguous, and uses the same terms as the text. Number, caption, and cite in text all illustrations and tables. Captions are short, for example:

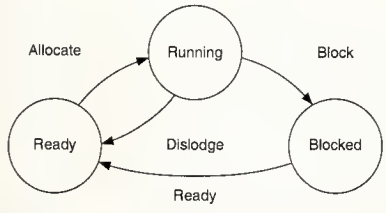


Figure 1. Task states and transitions. *(Copyright 1995 William Jones. Reprinted by permission.)*

Biographical sketch and photograph

Submit a photograph and biographical sketch of each author. Good-quality, black-and-white glossy photographs with good contract, preferably 7.5 × 12 cm (3 × 5 inches) in size, reproduce best. Limit biographical sketches to 75 words and include, in the following order: current positions and technical interests, prior professional experience and other important activities, education, professional affiliations, and current address. See a recent issue of *IEEE Micro* for examples.

Micro News

continued from p. 8

and Motorola, Inc. entered a joint venture to develop wireless electronics technology for remote and automated meter reading (RAMR). The new products will replace today's on-site visual inspections and hand-held data collection terminals.

The joint venture will take the form of an Atlanta, Georgia, design center that will also provide integrated solutions for water, gas, heat, and electricity utility meters on a global basis. Motorola will manufacture products, while Schlumberger provides metering products, marketing, sales, and customer services to utilities worldwide. Both companies will be equal owners of the company with equal representation on its board.

Electronic pen clipboards. PI Systems Corporation, developer of Infolio electronic pen clipboards, and Business Partner Solutions Inc., developer of AS/Messenger RadioPac wireless communications software, agreed to form a cooperative marketing partnership. The companies plan to develop wireless electronic pen clipboard services especially for health care providers, using Motorola InfoTAC and GE Ericcson Mobidem modems.

Designed to speed up the flow of error-free information to an organization's database, the real-time, pen-based interaction will supply database information to clinicians, whether in or out of the hospital environment. Expected benefits include reduced paperwork and billing cycles, real-time access to extensive patient information, and natural pen input for professionals.

Video/audio compression. Texas Instruments and C-Cube Microsystems recently announced an agreement to develop video and audio compression products. These products will be used in digital cable television and Direct Broadcast Satellite TV, HDTV, compact disc-based consumer video, and personal computer multimedia applications.

The agreement includes technology and product development exchanges between the two companies. It also provides each company with rights to develop derivative products of the other company's current and future MPEG and JPEG coder/decoder products. C-Cube will receive access to TI's advanced CMOS process technologies and production facilities. Specific applications for these products include Compact Disc-Interactive, CD-based karaoke players, CD-based video games, and new digital TV broadcast receivers.

Bill O'Meara of C-Cube stated, "With TI's marketing and production strength behind MPEG and JPEG, the digital video market is poised for explosive growth." Walden C. Rhines of TI's Semiconductor Group said, "The application of DSP techniques to consumer products promises to be a tremendous growth market for the semiconductor industry in the 1990s."

Traffic control. A neural network computer program originally developed to help military pilots deal with enemy threats may soon be used to ease traffic congestion, according to computer scientists at the Georgia Tech Research Institute in Atlanta. The TERMINUS traffic control program senses traffic conditions and regulates the operation of signal lights to optimize the flow of vehicles.

TERMINUS, short for Traffic Event Response and Management for Intelligent Navigation Utilizing Signals, runs on Sun Sparc and similar workstations. It represents each intersection as a neuron and each street segment between intersections as a neural interconnection.

The program displays an animated color map of streets, parking lots and the number of cars in them, and traffic conditions in potential problem areas. TERMINUS even provides a computer-generated sound of crashing vehicles to alert its operators to traffic accidents.

The initial application for the system simulated traffic conditions at the Atlanta Braves stadium to demonstrate

Micro bits

- Georgia Tech (xspice@gtrlgatech.edu) offers its **XSPICE simulator** through a no-cost license agreement and a $200 distribution charge. Useful when mixing system and analog simulations, the 1992 Unix SPICE extension in source code form is compatible with the original code.

- Dial 1-900-680-DEAL 24 hours/day, 7 days/week for **discount prices** on microcomputers, peripherals, printers, and fax machines. The P.C. Discount Shopper supplies biweekly updated product information available to callers via phone or fax. Each call costs $1.95/minute.

- A *Catalog of National ISDN Solutions for Selected NIUF Applications* selling for $44.50 describes 30 **ISDN applications** and the required equipment and services for building the applications. Contact the National Technical Information Service at (703) 487-4650 to order; specify PB 93-162881.

- X Business Group, a market research company, reports that the value of all **X specific products and services** sold worldwide increased 60 percent during 1992 to top $800 million. It also sets the worldwide installed base of X capable seats at over 2 million.

- The IEEE Computer Society has donated **"instant libraries"** to engineers and scientists at 30 sites in Eastern Europe and the former Soviet Union. Each library valued at $15,000 contains 200 authored books, reprint collections, and conference proceedings.

how signal light settings might be coordinated during special events. The next stage will create a hardware installation that can be integrated into an overall traffic management control system. Researchers will then join the system to a central traffic control computer and determine how it can accept and process data inputs from the sensors.

Larger applications will require the integration of complex geographic information systems into the work of TERMINUS.

Applications sought for manufacturing fellowships

US Commerce Secretary Ronald H. Brown announces the start-up of a program to place US engineers in Japanese manufacturing firms for up to one year. The goals of the Manufacturing Technology Fellowship project are to help US engineers learn more about—and then use—Japanese manufacturing practices and to promote long-term professional exchanges.

The program is accepting applications from US engineers sponsored by their companies. Fellowships will last about 15 months and include three months of intensive Japanese language and culture training in the US. Participants will learn about Kanban, just-in-time manufacturing, total quality control, and other techniques. Sixty Japanese companies will act as host organizations.

Potential candidates should contact project representatives at the US Department of Commerce by fax at (202) 482-4826. Applications must arrive by July 16, 1993.

VLSI Design 93 meets in Bombay

The Sixth International Conference on VLSI Design met in Bombay, India, January 3-6, with over 400 attendees from around the world. The conference was organized in cooperation with the ACM Special Interest Group on Design Automation, the IEEE Computer Society's Technical Committees on Design Automation and VLSI, and the IEEE Circuits and Systems Society. The conference also received support from the Department of Electronics of the Government of India.

Centering on chips, boards, and systems in the 90s, the conference began with the keynote address of Osamu Karatsu of NTT LSI Labs, Japan, on the "History and Future Directions of VLSI CAD and Design: A Japanese Perspective."

The two-day technical program consisted of 70 papers and 9 posters selected from a total of 186 submissions. Topics included logic synthesis, design for testability, physical design, testing, high-level synthesis, VLSI algorithms and architectures, parallel CAD, CAD frameworks, logic design, circuit design, and delay fault testing.

The conference also organized five one-day tutorials on topics such as FPGAs and DSP, exhibits of Indian CAD/CAE systems and VLSI/PCB design services, and two design contests for the Indian participants.

Next year's meeting will be held January 5-8 in Calcutta (see Call for Papers in March 1993 *Computer*). For information, contact Rochit Rajsuman (rajsuman@alpha.ces.cwru.edu).

PDA interest explored

According to a BIS Strategic Decisions survey, one third of the respondents indicated that they would buy a personal digital assistant, even though PDAs are not yet on the market. Surprisingly, price was not a key issue for potential buyers.

The PDA concept involves stylus-based computing, handwriting recognition, electronic organizing, word processing, spreadsheets, database managment, and other wireless communications functions.

Although the PDA market is often described as the future of pen computing, the survey showed that less than 25 percent of respondents prefer pen input; 54 percent preferred the keyboard.

Sales managers proved to be the best target market for PDAs. They are open to and interested in the concept and have the decision-making power to implement PDA use in their departments.

BIS Strategic Decisions, an international organization of industry analysts, produces a series of research reports on PDAs called *Emerging Markets for Personal Digital Assistants*. Interested parties can obtain pricing information from Robin Osborne, phone (617) 982-9500 or fax (617) 878-6650.

Literature

PC Design Guide includes a schematic summary sheet of chip sets available to designers of PC compatibles, and a list of manufacturers and phone numbers. *Annabooks, 15010 Avenue of Science, #101, San Diego, CA 92128-3421; (619) 673-0870; fax (619) 673-1432; $139.*

Engineers can now obtain the fourth edition of a 220-page handbook on making low-level measurements. Featured are step-by-step procedures, instructions, and a glossary of terms. *Keithley Instruments, Inc., 28775 Aurora Road, Cleveland, OH 44139; phone (216) 248-0400; fax (216) 248-6168; free.*

Need a how-to pocket guide to help you establish an open systems evnironment within your company? *A Very Useful Guide—Buying Open Systems* will help with the terminology and issues associated with purchasing open systems and the general process. *The 88open Consortium Ltd., 100 Homeland Court, Suite 800, San Jose, CA 95112; phone (408) 436-6600; fax (408) 436-0725; $8.*

Reader Interest Survey

Indicate your interest in this department by circling the appropriate number on the Reader Service Card.

Low 198 Medium 199 High 200

THE FOLLOWING INFORMATION IS AVAILABLE:

Contact the Publications Office;
to facilitate handling, please request by number.

- Membership application, student #203, others #202
- Publications catalog #201
- Compmail electronic mail brochure #194
- Technical committee list/application #197
- Chapters lists, start-up procedures #193
- Student scholarship information #192
- Volunteer leaders/staff directory #196
- IEEE senior member grade application #204
  (requires ten years practice and significant performance in five of those ten)

To check membership status or report a change of address, call the IEEE toll-free number, 1-800-678-4333. Direct all other Computer Society-related questions to the Publications Office.

IEEE COMPUTER SOCIETY®

A member society of the
Institute of Electrical and Electronics Engineers, Inc.

PURPOSE

The IEEE Computer Society advances the theory and practice of computer science and engineering, promotes the exchange of technical information among 100,000 members worldwide, and provides a wide range of services to members and nonmembers.

MEMBERSHIP

Members receive the acclaimed monthly magazine *Computer*, discounts, and opportunities to serve (all activities are led by volunteer members). Membership is open to all IEEE members, affiliate society members, and others interested in the computer field.

PUBLICATIONS AND ACTIVITIES

Computer. An authoritative, easy-to-read magazine containing tutorial and in-depth articles on topics across the computer field, plus news, conferences, calendar, interviews, and product reviews.

Periodicals. The society publishes eight magazines and seven research transactions. Refer to membership application or request information as noted at left.

Conference Proceedings, Tutorial Texts, Standards Documents. The Computer Society Press publishes more than 100 titles every year.

Standards Working Groups. More than 100 of these groups produce IEEE standards used throughout the industrial world.

Technical Committees. More than 30 TCs publish newsletters, provide interaction with peers in specialty areas, and directly influence standards, conferences, and education.

Conferences/Education. The society holds about 100 conferences each year and sponsors many educational activities, including computing science accreditation.

Chapters. Regular and student chapters worldwide provide the opportunity to interact with colleagues, hear technical experts, and serve the local professional community.

OMBUDSMAN

Members experiencing problems — magazine delivery, membership status, or unresolved complaints — may write to the ombudsman at the Publications Office.

EXECUTIVE COMMITTEE

President: James H. Aylor*
University of Virginia
Thornton Hall/Electrical Engineering
Charlottesville, VA 22903
Phone: (804) 924-6100
Fax: (804) 924-8818
E-mail: jha@virginia.edu

President-Elect: Laurel V. Kaleda*
Past President: Bruce D. Shriver*

VP, Technical Activities: Joseph Boykin (1st VP)*
VP, Conferences and Tutorials: Anneliese Von Mayrhauser (2nd VP)*
VP, Educational Activities: Gerald L. Engel†
VP, Membership Activities: Fiorenza C. Albert-Howard*
VP, Press Activities: Ronald G. Hoelzeman†
VP, Publications: Barry W. Johnson†
VP, Standards Activities: Gary S. Robinson†

Secretary: Mario R. Barbacci*
Treasurer: Michael Evangelist†
IEEE Division V Director: Bill D. Carroll†
IEEE Division VIII Director: V. Tom Rhyne†
Executive Director: T. Michael Elliott†

*voting member of the Board of Governors
†nonvoting member of the Board of Governors

BOARD OF GOVERNORS

Term Expiring 1993:
Fiorenza C. Albert-Howard, Jon T. Butler,
Michael C. Mulder, Yale N. Patt, Benjamin W. Wah,
Ronald Waxman, Akihiko Yamada

Term Expiring 1994:
Mario R. Barbacci, L. Felipe Cabrera, Wolfgang K. Giloi, Guylaine M. Pollock, John P. Riganati, Ronald D. Williams, Thomas W. Williams

Term Expiring 1995:
Fletcher J. Buckley, Doris L. Carver, Elliot J. Chikofsky,
Joanne E. DeGroat, Michael J. Flynn,
Mary Jane Irwin, Grace C.N. Wei

Next Board Meeting
November 21, 1993, 8:30 a.m.
Santa Clara Marriott, Santa Clara, Calif.

SENIOR STAFF

Executive Director: T. Michael Elliott
Publisher: H. True Seaborn
Director, Conferences and Tutorials: Anne Marie Kelly
Director, Finance and Information Services: Deborah Rafal
Director, Board and Administrative Services: Violet S. Doan
Assistant to the Executive Director: Sandra K. Pfau

COMPUTER SOCIETY OFFICES

Headquarters Office
1730 Massachusetts Ave. NW
Washington, DC 20036-1903
Phone: (202) 371-0101
Fax: (202) 728-9614
E-mail: hq.ofc@compmail.com

Publications Office
10662 Los Vaqueros Cir.
PD Box 3014
Los Alamitos, CA 90720-1264
Membership and General Information:
(714) 8218380
membership@compmail.com
Publication Orders: (800) 272-6657
Fax: (714) 821-4010
E-mail: cs.books@compmail.com

European Office
13, Ave. de L'Aquilon
B-1200 Brussels, Belgium
Phone: 32 (2) 770-21-98
Fax: 32 (2) 770-85-05
E-mail: euro.ofc@compmail.com

Asia/Pacific Office
Doshima Building
2-19-1 Minami-Aoyama, Minato-ku
Tokyo 107, Japan
Phone: 81 (3) 3408-3118
Fax: 81 (3) 3408-3553
E-mail: tokyo.ofc@compmail.com

IEEE OFFICERS

President: Martha Sloan
President-Elect: H. Troy Nagle
Past President: Merrill W. Buckley, Jr.

Secretary: Souguil J.M. Ann
Treasurer: Theodore W. Hissey, Jr

VP, Educational Activities: Edward A. Parrish
VP, Professional Activities: Charles K. Aledxander
VP, Publication Activities: Helen M. Wood

VP, Regional Activities: Luis T. Gandia
VP, Standards Activities: Wallace S. Read
VP, Technical Activities: Donald M. Bolle

IEEE COMPUTER SOCIETY PRESS

SOFTWARE REENGINEERING

edited by Robert S. Arnold

This tutorial covers a wide variety of interesting software reengineering approaches and introduces its themes, strategies, technology, and risks. Explores and defines software reengineering concepts and processes, tools and techniques, capabilities and limitations, risks and benefits, and research possibilities.

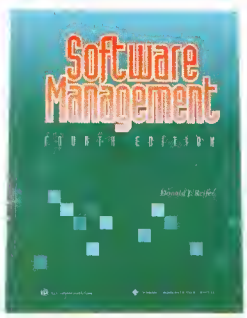
Sections: Context and Definition, Business Process Reengineering, Strategies and Economics, Reengineering Experience and Evaluation, Technology for Reengineering, Data Reengineering and Migration, Source Code Analysis, Software Restructuring and Translation, Documenting Existing Programs, Reengineering for Reuse, Reverse Engineering and Design Recovery, Object Recovery, Knowledge-Based Program Analysis.

688 pages. ISBN 0-8186-3272-0. April 1993. Hardcover.
Catalog No. 3272-01 — List Price $79.00 Members $64.00

SOFTWARE MANAGEMENT, 4th Edition

edited by Donald J. Reifer

The tutorial includes both original material and reprints that amplify related management theories, concepts, tools, and techniques and provide guidelines to improve the practice. Its text also includes coverage of process assessment, metrics, and risk management topics. Provides managers with an in-depth study of the available technology for managing the process, product, and people involved in software development and maintenance.



Sections: Software Process, Project Management, Planning Fundamentals, Organizing for Success, Staffing Essentials, Direction Advice, Visibility and Control, Risk Management, Metrics and Measurement, Software Engineering Technology Transfer, Support Material.

664 pages. ISBN 0-8186-3342-5. April 1993. Hardcover.
Catalog No. 3342-01 — List Price $79.00 Members $64.00

GROUPWARE: Software for Computer-Supported Cooperative Work

edited by David Marca and Geoffrey Bock

This book focuses on the development of new software that enhances cooperation and augments human capability. It is a collection of distinctions, approaches, methods, and examples that are transforming the development of computer systems for groups. Describes key topics such as user interface technologies, ways to design software to enhance human capabilities, methods and applications to coordinate human activity, and the effectiveness of groupware. The text also concentrates on the design of software to fit the way groups interact in specific work situations.

Sections: Introduction, Groups and Groupware, Conceptual Frameworks, Design Methods, Enabling Technologies - System-Related, Enabling Technologies - UI-Related, Computer-Supported Meetings, Bridging Time and Space, Coordinators, What Makes Systems Effective, Bibliography, Index.

592 pages. ISBN 0-8186-2637-2.
August 1992. Hardcover.
Catalog No. 2637-01
List Price $75.00
Members $45.00

COMPUTER-AIDED SOFTWARE ENGINEERING (CASE), 2nd Edition

edited by Elliot Chikofsky

This new edition presents new information on CASE technology, and examines its present state, how its concepts have fared over time, and how it looks as a technology for the future. Features more than 35% new papers and combines the latest papers in the field with background articles that allow the reader to see how the field is evolving.

Sections: CASE Environments and Tools: Overview, Role of Assistants and Expert System Technology in CASE, Evolution of Software Development Environment Concepts, Role of Prototyping in CASE, Role of Data Browsing Technology in CASE, Tailoring Environments (Extension, Meta-Specification, Generation), Issues of Evaluating Tools and Managing CASE.

184 pages. ISBN 0-8186-3590-8.
January 1993. Softcover.
Catalog No. 3590-05
List Price $35.00
Members $25.00

COMING SOON

SIMULATION VALIDATION
by Peter L. Knepell and Deborah C. Arang
Catalog # 3512-04

SOFTWARE ENGINEERING: A European Perspective
edited by Richard H. Thayer and Andrew D. McGettri
Catalog # 2117-01

***Readings in* REAL-TIME SYSTEMS**
edited by Y. H. Lee and C. M. Krishna
Catalog # 2997-01

***Readings in* DISTRIBUTED COMPUTING SYSTEMS**
edited by Thomas L. Casavant and Mukesh Singhal
Catalog # 3032-01

IEEE COMPUTER SOCIETY
10662 Los Vaqueros Circle
Los Alamitos, CA 90720

Call toll-free 1-800-CS-BOOKS
in CA (714) 821-8380
or FAX (714) 821-4641